Time-Dependent Analysis of Attacks

Florian Arnold[1], Holger Hermanns[2], Reza Pulungan[3], and Mariëlle Stoelinga[1]

[1] Formal Methods & Tools Group, Department of Computer Science
University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
{f.arnold,m.i.a.stoelinga}@utwente.nl
[2] Dependable Systems and Software, Saarland University,
66123 Saarbrücken, Germany
hermanns@cs.uni-saarland.de
[3] Jurusan Ilmu Komputer dan Elektronika, Universitas Gadjah Mada, Indonesia
pulungan@ugm.ac.id

Abstract. The success of a security attack crucially depends on time: the more time available to the attacker, the higher the probability of a successful attack; when given enough time, any system can be compromised. Insight in time-dependent behaviors of attacks and the evolution of the attacker's success as time progresses is therefore a key for effective countermeasures in securing systems.

This paper presents an efficient technique to analyze attack times for an extension of the prominent formalism of attack trees. If each basic attack step, *i.e.*, each leaf in an attack tree, is annotated with a probability distribution of the time needed for this step to be successful, we show how this information can be propagated to an analysis of the entire tree. In this way, we obtain the probability distribution for the entire system to be attacked successfully as time progresses. For our approach to be effective, we take great care to always work with the best possible compression of the representations of the probability distributions arising. This is achieved by an elegant calculus of acyclic phase type distributions, together with an effective compositional compression technique. We demonstrate the effectiveness of this approach on three case studies, exhibiting orders of magnitude of compression.

1 Introduction

Computer security attacks on traditional IT-systems as well as on modern IT-enabled systems, such as cars, pacemakers, or power grid infrastructure, are on the rise. Successful attacks have a certain structure and timing, and one of the dominant problems in preventing attacks is that the security engineers fail to properly predict the potential angle and timing of an attack.

In a recent article [1] Basin and Capkun argue that there is a lot to learn from the study of successful attacks. In particular, it can help in refining the attacker model, so as to understand the potential attack angles better, and thus arrive at ever more effective countermeasures. In this context, security engineers and product managers are facing the practical challenge of limited resources. In the

M. Abadi and S. Kremer (Eds.): POST 2014, LNCS 8414, pp. 285–305, 2014.

end, decisions have to be made about how much to spend on security, and where to invest their budget. Making well-informed choices requires insight in attacks: which attacks seem more likely than others? How much time might they take to succeed?

There is a growing awareness that the mathematical foundations of such quantitative aspects of security are worth to be better investigated [2]. Concretely, the quantitative analysis of attacks can yield valuable insights in aspects like (1) system parameters: which parameters influence attack times and probabilities most? (2) what-if scenarios, e.g., what if attack probabilities increase with two orders of magnitude? (3) design alternatives. This paper contributes to this research strand, and it does so in the context of a prominent practical attack modelling formalism, attack trees.

Attack trees (AT) were coined by Schneier [3], as a means to describe, document, brainstorm, and analyze system security. Over the last decades indeed a wide range of techniques have been developed to analyze costs, probability, effort, etc., associated to a successful attack [4–7].

This paper studies the probability of a successful attack as time advances, *i.e.*, the probability distribution of attack times: given a time bound t, what is the probability that the system is successfully compromised within time t? Our probabilistic timed analysis of attacks is conservatively extending earlier time-abstract analyses [3, 5, 7, 8]. The latter only considered the probability whether or not an attack eventually could take place, while we evaluate the success probability as a function of time.

To represent probabilistic timed behaviour, we use *acyclic phase-type* (APH) distributions. APH distributions are a distingushed class of probability distributions, as they can be used to approximate any other probability distribution with arbitrary precision; and—as we fruitfully exploit in this paper—allow for very compact representations. Furthermore, effective fitting techniques exist to derive APH distributions from statistical data [9, 10]. We therefore assume that each leaf of an attack tree is annotated with an APH distribution representing the time needed for this step to be successful. Of course, one may argue that it is difficult to get realistic data about the timing of these attack steps—apart from the attacker anyway behaving notoriously unpredictably—and that therefore, the outcomes of this quantitative security analyses should not be trusted. Still, the benefit of the approach we propose lies in the possibility to easily pose and effectively evaluate 'what if' questions, understand system parameters, and to study design alternatives at the push of a button. Especially the 'what if' approach gives a way to understand the sensitivity of the system with respect to different leaf distributions.

At the core of the paper's contribution lies a compositional attack tree semantics that maps on APH distributions, together with a compositional compression algorithm. Here, compositionality refers to the possibility to weave the compression into the compositional construction of the APH distribution associated with the entire graph. This keeps the representations as small as possible. The compression exploits Laplace domain properties in a symbolic and effective manner.

Three case studies exemplify the effectiveness of this approach, among them a study of the Stuxnet attack, and a novel and large industrial case. The case studies are carried out with a full-fledged implementation of the approach we present.

Related Work. Weiss's threat logic trees [11] and Amoroso's threat trees [12] were the first formal models to represent attacks for security analysis. In 1999, Schneier introduced the notion of attack trees [3]. Since then a colorful variety of new formalisms has evolved. These variations can be classified in *static* models, which do not take the evolution of time into account, and *dynamic* models which can express timed behavior such as sequencing. A more detailed overview and classification can be found in [13].

Static models have been rigorously formalized by Mauw and Oostdijk [14]. Multi-parameter attack trees [4] were introduced to process different parameters in parallel. The idea to shift the focus from the attacker's to the defender's point of view was captured by the introduction of defense trees [15], [16] and attack-countermeasure trees [6]. While these tree-based models are evaluated by a bottom-up analysis [5], graph-based formalisms suggested by Sheyner et al. [17] can be analyzed with model checking.

In the field of dynamic attack models, most formalisms are graph-based, for instance compromise graphs developed by McQueen [18]. Distinct attacker preferences were introduced in [19]. More expressive approaches use Petri nets to model intrusion detection as attack nets [20]. A tree-based formalism has only recently been introduced by Pièvre-Cambacédès and Bouissou [21, 22] which enables the modeling of sequences within complex attack scenarios.

2 Preliminaries

Random Variables. A real-valued *random variable* is a function $X : \Omega \to \mathbb{R}$ that assigns a real value to each outcome of a stochastic experiment; in our case, X describes the time it takes until a system is successfully attacked. Then $\mathbb{P}(X \leq t)$ denotes the probability that X has a value less than or equal to t; in our case the probability that a successful attack occurs within time t. The function $F : \mathbb{R} \to [0, 1]$ given by

$$F(t) = \mathbb{P}(X \leq t)$$

is called the *cumulative distribution function (CDF)* of X; and $X \sim F$ denotes that X has CDF F. (Note that, in many cases, $\mathbb{P}(X = t) = 0$. Therefore one considers the cumulative probability $\mathbb{P}(X \leq t)$.) We denote by $\mathcal{F}$ the class of all cumulative distribution functions.

Well-known examples are the (negative) exponential distribution whose CDF is given by

$$\mathbb{P}(X \leq t) = 1 - e^{-\lambda t}, \quad \text{for any } t \in \mathbb{R}^+.$$

This distribution has a parameter $\lambda > 0$, called rate, which determines the "speed" with which the probability grows. We write $X \sim \exp(\lambda)$ if X has an exponential distribution.

Apart from its CDF, a random variable can also be characterized by its *probability density function (PDF, or density for short)* f, which is the derivative of the CDF

$$\mathbb{P}(X \leq t) = \int_{-\infty}^{t} f(x)dx.$$

Thus, the PDF of the exponential distribution is given by $f(t) = \lambda e^{-\lambda t}$.

Furthermore, it is important to realize that, given a real-valued random variable X, and a real-valued function $f : \mathbb{R} \to \mathbb{R}$, $f(X)$ is again a random variable. Given several random variables $X_1, X_2, \ldots X_n$ and a function $g : \mathbb{R}^2 \to \mathbb{R}$, we have by repetitive application that $g(X_1, g(X_2, \ldots, g(X_{n-1}, X_n)))$ is a random variable. Below, we will heavily consider random variables $X_1 + X_2 + \ldots + X_n$, $\max(X_1, X_2, \ldots X_n)$, and $\min(X_1, X_2, \ldots X_n)$. If we take the sum of k independent random variables each governed by an exponential distribution with parameter λ, then we obtain the *Erlang distribution* with parameters k, λ. Its density $f(x) = \frac{\lambda^k}{(k-1)!} x^{k-1} e^{-\lambda x}$ is the so-called *convolution* of the PDFs of exponential distributions. We write $X \sim \mathrm{erl}(k, \lambda)$ if X is Erlang-distributed.

Acyclic Phase-Type Distributions. Phase-type distributions are represented by the time needed to reach a final state in a *continuous-time Markov chain* (CTMC). If this Markov chain is acyclic, then we speak about an *acyclic phase-type distribution (APH).* APHs consitute a very prominent class of probability distributions: they subsume the exponential and Erlang distributions. Notably, APH distributions are topologically dense [23]. This implies that any continuous distribution can be approximated arbitrarily closely by an APH distribution or a PH distribution. Very effective tools exist that compute tight approximations of arbitrary distributions by small APH distributions or fit an APH distribution to measurements, *i.e.*, given a set of empirical data, they can compute the APH distribution that matches most closely [9, 10, 24]. Moreover, as we show below, APH distributions are closed under summation, maximum, and minimum and allow for drastic compression techniques. All this makes the APH distributions a suitable class to model time-dependent behavior of attack trees.

A CTMC is a tuple $\mathcal{M} = (\mathcal{S}, \mathbf{Q}, \boldsymbol{\pi})$, where $\mathcal{S} = \{s_1, s_2, \cdots, s_n, s_{n+1}\}$ is a countable set of states, $\mathbf{Q} : (\mathcal{S} \times \mathcal{S}) \to \mathbb{R}$ is a so-called infinitesimal *generator matrix*, and $\boldsymbol{\pi} : \mathcal{S} \to [0, 1]$ is the initial probability distribution on $\mathcal{S}$. Intuitively, for any two states $s, s' \in \mathcal{S}$, $\mathbf{Q}(s, s')$ specifies the *rate* of the transition from s to s'. This means that the probability that a state change occurs from s to s' within t time units is $1 - \exp(-\mathbf{Q}(s, s')t)$. By definition $\mathbf{Q}(s, s') \geq 0$ for all $s \neq s'$, and $\mathbf{Q}(s, s) = -\sum_{s \neq s'} \mathbf{Q}(s, s')$. The negative of the diagonal value, $E(s) = -\mathbf{Q}(s, s)$, is called the *exit rate* of state s.

If state s_{n+1} is absorbing (*i.e.*, $E(s_{n+1}) = 0$) and all other states s_i, for $1 \leq i \leq n$, are transient (*i.e.*, there is a nonzero probability that s_i will never be visited once it is left), the generator matrix of the CTMC can be written as

$$\mathbf{Q} = \begin{bmatrix} \mathbf{A} & \boldsymbol{A} \\ \mathbf{0} & 0 \end{bmatrix}.$$

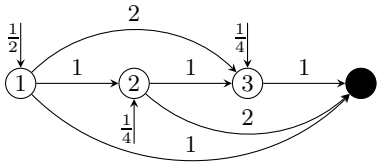


Fig. 1. An acyclic PH distribution

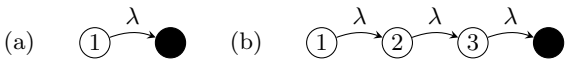


Fig. 2. Graphical representation of (a) an exponential distribution and (b) an Erlang distribution with 3 phases

In this paper we consider CTMCs with acyclic graph structure, or, from the matrix perspective, with $\mathbf{A}$ being an upper triangular matrix. Fig. 1 shows an example of such an acyclic CTMC. The probability distribution of the time until the absorbing state is reached in such a CTMC is called an APH distribution [25]. Together with the initial probability vector at transient states $\boldsymbol{\alpha}$, matrix $\mathbf{A}$ completely characterizes the APH distribution under consideration. The pair $(\boldsymbol{\alpha}, \mathbf{A})$ is called the *representation* of the APH distribution. The CDF of the APH distribution, which represents the probability distribution of the time to absorption mentioned above, is given by

$$F(t) = \mathbb{P}(X \leq t) = 1 - \boldsymbol{\alpha} e^{\mathbf{A}t} \mathbf{1}, \tag{1}$$

where $\mathbf{1}$ is a vector of proper size whose components are all 1.

The simplest APH distributions are formed by the family of exponential and Erlang distributions; see Fig. 2 for their graph-based APH representations.

Three Stochastic Operations. We consider three operations on continuous probability distributions, since they have very intuitive correspondences with the operators on attack trees. Let X_1 and X_2 be two independent random variables with distribution functions $F_1(t)$ and $F_2(t)$, respectively; and let the random variables $X_{\text{con}} = X_1 + X_2$, $X_{\max} = \max\{X_1, X_2\}$, and $X_{\min} = \min\{X_1, X_2\}$ be the summation (convolution), maximum, and minimum, respectively, of X_1 and X_2. The random variables X_{con}, $X_{\max}$, and $X_{\min}$, have CDFs $F_{\text{con}}(t) = \int_0^t F_1(t - x)F_2(x)dx$, $F_{\max}(t) = F_1(t)F_2(t)$, and $F_{\min}(t) = 1 - (1 - F_1(t))(1 - F_2(t))$, respectively.

Acyclic phase-type distributions are closed under these three operations. Given APH distributions PH_1 and PH_2, we use $\text{con}(\mathsf{PH}_1, \mathsf{PH}_2)$, $\max(\mathsf{PH}_1, \mathsf{PH}_2)$, and $\min(\mathsf{PH}_1, \mathsf{PH}_2)$ to denote the convolution, maximum, and respectively minimum of the two APH distributions. The following theorem provides the recipe to obtain the representation of the convolution, maximum, and minimum, given the representations of the two constituent APH distributions.

Theorem 1. *[25, Theorem 2.2.9] Let* $(\boldsymbol{\alpha}, \mathbf{A})$ *and* $(\boldsymbol{\beta}, \mathbf{B})$ *be the representations of PH distributions* $F(t)$ *and* $G(t)$ *of size* m *and* n*, respectively. Then*

(a) their convolution *is a PH distribution with representation* $(\boldsymbol{\delta}, \mathbf{D})$ *of size* $m+n$*, where*

$$\boldsymbol{\delta} = [\boldsymbol{\alpha}, \alpha_{m+1}\boldsymbol{\beta}] \quad \textit{and} \quad \mathbf{D} = \begin{bmatrix} \mathbf{A} & \boldsymbol{A\beta} \\ \mathbf{0} & \mathbf{B} \end{bmatrix}.$$

(b) their maximum *is a PH distribution with representation* $(\boldsymbol{\delta}, \mathbf{D})$ *of size* $mn+m+n$*, where*[1]

$$\boldsymbol{\delta} = \begin{bmatrix} \boldsymbol{\alpha} \otimes \boldsymbol{\beta}, & \beta_{n+1}\boldsymbol{\alpha}, & \alpha_{m+1}\boldsymbol{\beta} \end{bmatrix} \quad \textit{and} \quad \mathbf{D} = \begin{bmatrix} \mathbf{A} \oplus \mathbf{B} & \mathbf{I}_A \otimes \boldsymbol{B} & \boldsymbol{A} \otimes \mathbf{I}_B \\ \mathbf{0} & \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B} \end{bmatrix}.$$

(c) their minimum *is a PH distribution with representation* $(\boldsymbol{\delta}, \mathbf{D})$ *of size* mn*, where*

$$\boldsymbol{\delta} = \boldsymbol{\alpha} \otimes \boldsymbol{\beta} \quad \textit{and} \quad \mathbf{D} = \mathbf{A} \oplus \mathbf{B}.$$

3 Attack Trees

Attack trees establish an intuitive model to systematically describe possible attack scenarios on a system and thereby form the basis for a threat analysis.

Attack Tree Syntax. The graphical representation of an AT is a tree. The root node of the tree identifies the goal of the attacker within the considered scenario. This goal can be achieved by executing a series of basic attack steps (BAS) which are encoded as leaves of the AT. A BAS describes one action of the attacker to exploit the system's vulnerabilities which can not be refined into finer steps. Complex attack scenarios are described by composing the involved BASs. The syntactic tool to express this composition in an AT are so-called gates. Most AT formalisms use AND gates and OR gates to describe conjunctive and disjunctive composition respectively [3,14]. Additionally, we introduce SEQ gates to model time dependencies between BASs. This gate is described in more detail in the sequel. We use the standard distinctive shapes to visualize AND gates and OR gates. The graphical representation of an SEQ gate is an AND gate with a horizontal arrow pointing in the direction of progressing time.

The syntax and semantics of ATs allow the leaves to be annotated with any CDF in $\mathcal{F}$, but our analysis techniques exploit the fact that we work with APH distributions.

Definition 1 (AT gates and elements). *An* AT gate *is one of* AND*,* OR *or* SEQ*. We denote by* $\mathcal{G}$ *the set of gates. An* AT element *is either a gate, or a CDF. The set of AT elements* $\mathcal{E}$ *is given by* $\mathcal{E} = \mathcal{G} \cup \mathcal{F}$*.*

[1] $\otimes$ and $\oplus$ denote the Kronecker product and sum operators, respectively. See for instance [25].

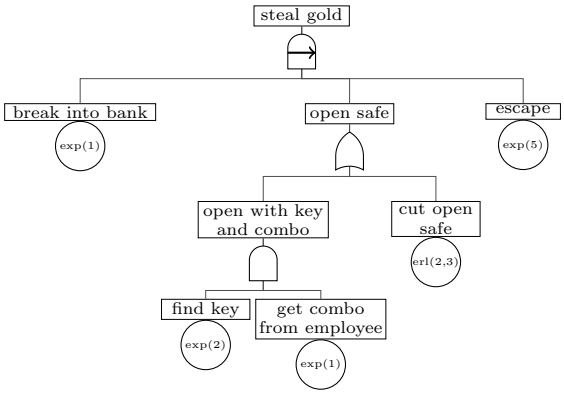


Fig. 3. Attack tree for the scenario 'steal gold from a bank'

In the sequel, given a set X, we let X^* denote the set of all sequences, also called lists, over X. For a list $x \in X^*$, let $|x|$ denote its length; $(x)_i$ is the i-th element of x. Informally, an attack tree is a directed acyclic graph whose leaves are labeled by an element of $\mathcal{F}$; all other nodes are gates, being either AND, OR or SEQ.

Definition 2 (Attack tree syntax). *An attack tree $\mathcal{A}$ is a graph (V, child, r, l), where*

- *V is a finite set of* vertices.
- $\text{child} : V \to V^*$ *assigns to each vertex a list of* input vertices.
- *We define the set of* edges *of $\mathcal{A}$ by $E_{\mathcal{A}} = \{(v, w) \in V^2 \mid \exists i \,.\, w = (\text{child}(v))_i\}$. We require that $(V, E_{\mathcal{A}})$ is acyclic with a single root $r \in V$. All vertices have to reach r.*
- *We denote by L the* leaves *of $(V, E_{\mathcal{A}})$, so that $L = \{v \in V \mid |\text{child}(v)| = 0\}$.*
- *$\ell : V \to \mathcal{E}$ is a labeling function that assigns to each vertex an AT element such that*
  - *Each leaf in A is annotated with a CDF: $\ell(v) \in \mathcal{F}$, for all $v \in L$;*
  - *All other vertices are annotated with gates: $\ell(v) \in \mathcal{G}$ for all $v \in V \backslash L$.*

Example 1. The attack tree in Fig. 3 models the attack scenario to rob gold from a bank. To do so, the attacker first has to get into the bank before he can try to open the safe. This time dependency is modeled with a SEQ gate. The attacker has two possible ways to open the safe; he can either unlock it or cut it open; hence, the OR gate. To unlock the safe, the attacker needs to find a key and obtain the access combo from an employee. As there is no time dependency between these two steps, we compose them with an AND gate. After opening the safe, the attacker has to make for a safe escape with his booty. The distribution of the time to execute each BAS is modeled as exponential or Erlang distribution.

Timed Probabilistic Semantics of Attack Trees. In this section we define the semantics of the AT elements. We aggregate the CDFs of the BASs along the gates to obtain a single CDF for an AT $\mathcal{A}$, making use of the operators introduced above. Therefore, the semantics of each AT element, denoted by $[\![.]\!]$, is a CDF. We aim to derive $[\![r]\!]$ for the root node r. We further define $[\![\mathcal{A}]\!] = [\![r]\!]$. Let in the following $v \in V(A)$ be a node of $\mathcal{A}$ with inputs $v_1, \ldots, v_n$.

BAS. Let v be a leaf, so $\ell(v) \in \mathcal{F}$. It is then annotated with a CDF that represents the distribution of the time to execute this attack step. We formalize this interpretation by defining the semantics of a leaf node as its annotated CDF, *i.e.*, $[\![v]\!] = L(v)$.

AND gate. Let vertex v be labeled with AND, $\ell(v) = \text{AND}$. Then the attack step represented by v is completed by the attacker, once each of the steps represented by its input vertices are completed. This corresponds to the latest time a step represented by the input vertices is completed which in turn is expressed by the maximum over the CDFs of its inputs, in other words $[\![\text{AND}(v_1, \ldots, v_n)]\!] = \max\{[\![v_1]\!], \ldots, [\![v_n]\!]\}$.

OR gate. Let $\ell(v) = \text{OR}$. Then the attack step represented by vertex v is completed, when at least one of the steps represented by its input vertices is completed. This corresponds to the earliest time at which an attack step represented by any of the input vertices is completed. Analogous to the AND gate we thus define its semantics using the minimum of the CDFs: $[\![\text{OR}(v_1, \ldots, v_n)]\!] = \min\{[\![v_1]\!], \ldots, [\![v_n]\!]\}$.

SEQ gate. Finally, let $\ell(v) = \text{SEQ}$. The semantics of a SEQ gate is similar to that of an AND gate in the sense that the attack step represented by it is only achieved if all the steps represented by its input vertices are completed. In addition, it expresses a causal dependency of BASs that induces a temporal order: attack steps can only be executed in succession: A step commences at the moment another step is successfully completed. The time at which the step represented by vertex v is achieved corresponds to the sum of the times required to complete each of the attack steps represented by its input vertices. As these times are random variables distributed according to CDFs, we use the convolution operation to define the semantics of the SEQ gate as $[\![\text{SEQ}(v_1, \ldots, v_n)]\!] = \text{con}\{[\![v_1]\!], \ldots, [\![v_n]\!]\}$. The SEQ gate is a novelty in attack modeling, its semantics is inspired by the 'trigger' element in [21].

The CDF $[\![\mathcal{A}]\!]$ corresponding to the entire attack tree $\mathcal{A}$ is derived by composing the CDFs in the leaves with maximum, minimum, and convolution operations along the tree structure.

4 Efficient Analysis of Attack Trees

This section shows how we can efficiently analyze attack times for ATs whose leaves are annotated with APH distributions, based on clever representations of these. In general, the CTMC representation of an APH distribution is not unique, and two representations of one APH distribution can differ drastically in size. In practice, there is a need to have the smallest possible representation. This

holds in particular when applying the minimum, maximum and sum operators, since these yield exponential blow ups of the CTMC representations. To tackle this problem, we discuss two important representations: the ordered bidiagonal representation, and the Cox representation.

APH Representations. The size of an APH representation is the dimension of $\mathbf{A}$. Notably, any APH distribution has infinitely many representations of different sizes. An *(acyclic) minimal representation* of an APH distribution is an APH representation with the least possible number of states.

There are two different canonical forms of APH representations, ordered bidiagonal and Cox forms, each with a simple and easy-to-understand structure. Each of them is "canonical" in the sense that it (if viewed as a graph) is unique up to isomorphism. Every APH representation can be transformed into either of them without altering its stochastic behavior, *i.e.*, its distribution.

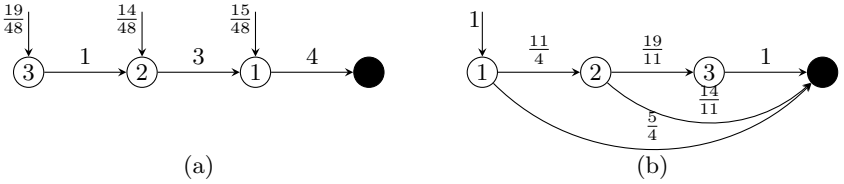


Fig. 4. (a) An ordered bidiagonal and (b) a Cox representations of the APH representation from Fig. 1

Ordered Bidiagonal. Fig. 4(a) depicts an *ordered bidiagonal representation*. Such representation has a simple structure: the states are ascendingly ordered by their exit rates, each of them has only one transition to its neighbour, and the initial distributions spans the entire state space. An efficient algorithm, called the *spectral polynomial algorithm* (SPA) [26], can be used to construct the ordered bidiagonal representation of any given APH representation. SPA has complexity $\mathcal{O}(n^3)$, where n is the size of the given APH representation.

Cox. A *Dirac* distribution is a probability distribution that assigns full probability to a single outcome. Consider the representation depicted in Fig. 4(b). Here, every state, apart from the absorbing state, has a transition to the next state, possibly a transition to the absorbing state, and no other transitions. If the representation has descending exit rates and a Dirac initial distribution to the highest exit rate state, then it is called a *Cox representation*. The name is due to David R. Cox [27], who coined this representation.

Once an ordered bidigonal representation is obtained, transforming it to the associated Cox representation can be performed with complexity $\mathcal{O}(n)$ [28], where n is the number of states in the resulting representation.

Size Compression. Given the three operations introduced in Section 2, Theorem 1 is the basis for constructing complex APH representations from simpler ones. However, in practice we often face an explosion in size, especially rooted

in the fact that the maximum and minimum are basically cross-product constructions. Hence, they grow as the product of their component sizes, while the convolution grows as their sum. This phenomenon is not uncommon especially for concurrent system representations. Luckily, we have an effective means to compress the resulting sizes considerably in almost all cases. This is rooted in a polynomial-time algorithm [29] that compresses the size of any APH representations. Due to space constraints, we only provide a brief summary of the functioning of this algorithm below. For an exhaustive discussion of the algorithm and its properties, the interested reader is referred to [29].

Given an arbitrary APH representation $(\boldsymbol{\alpha}, \mathbf{A})$ as input, the algorithm returns an ordered bidiagonal representation, denoted $\mathsf{Red}(\boldsymbol{\alpha}, \mathbf{A})$, having the same PH distribution, with a representation size being at most the size of the original one. The compression achievable goes beyond concepts like lumpability [30], since the algorithm exploits properties of the Laplace-Stieltjes transform. In very brief terms, the algorithm checks, for each matrix row and column certain dependencies in the matrix of its ordered bidiagonal representation of the distribution (obtained by running the SPA algorithm a priori). These dependencies are expressible in terms of linear equation systems. If satisfied, a state can be identified as being removable, and is subsequently removed from the representation. This process is then iterated until no further state is identified as being removable.

Overall, the algorithm has complexity $\mathcal{O}(n^3)$, where n is the number of states of the original representation. It can be applied to arbitrary APH distributions. In fact, this can turn an exponential growth (in the number of composed components) of the matrix size into a linear growth which is almost certainly a minimal APH representation [29].

Implementation. We have implemented a tool to generate and manipulate APH representations, together with the compression algorithm in a toolsuite called APHzip. The tool accepts as input an expression written in a prefix notation The expressions follow the grammar

$$P ::= \exp(\lambda) \mid \mathrm{erl}(k, \lambda) \mid \mathrm{con}(P, P) \mid \max(P, P) \mid \min(P, P) \mid \mathrm{cox}(\mu, \lambda, P),$$

where $\lambda \in \mathbb{R}_+$ and $\mu \in \mathbb{R}_{\geq 0}$ are *rates*, and $k \in \mathbb{Z}_+$. Here, $\exp(\lambda)$ and $\mathrm{erl}(k, \lambda)$ represent exponential and Erlang distributions; these are the building blocks of more complex APHs constructed by using operators convolution (con), maximum (max), and minimum (min). $\mathrm{cox}(\mu, \lambda, P)$ is an operator used to produce Cox representations. This operator semantically works as follows: given an APH P, $\mathrm{cox}(\mu, \lambda, P)$ is a new APH obtained by adding a new state having a transition with rate μ to the absorbing state and a transition with rate λ to P. Repeated application of this operator enables us to produce any Cox representation, and hence represent any APH distribution (and hence approximate any distribution with arbitrary precision) as APHzip expressions.

The user-perceived functionality of APHzip is simple. It takes an expression, compresses it, and returns the resulting compressed APH representation, either as a nested $\mathrm{cox}(\cdot)$-expression, or as a file containing the result as a simple list

of transitions. For end users experimenting with attack trees and similar formalisms, APHzip is wrapped in a web-based interface accessible at the address `http://depend.cs.uni-saarland.de/tools/aphzip`.

Phase-Type Fitting for a Basic Attack Step. Generally, there are two ways to obtain the CDF for a BAS: 1) with historical data; or 2) on the basis of expert opinion. If empirical data is available, that data is provided as input to a fitting tool, preferably G-Fit [10]. G-Fit then produces the best matching CDF, represented by a hyper-Erlang distribution. Since hyper-Erlang distributions are a subset of the family of APH, we already have a convenient input format for our analysis. A second possibility is that experts estimate the mean time t to execute this BAS and then use $\exp(1/t)$ as APH representation. This is justified by the fact that the exponential distribution is the "most random" of all distributions with a given mean, in the sense that it has maximal entropy. Indeed, other attack models use the exponential distribution as a default choice [21, 31].

Time-Dependent Analysis. In this section we use the above concepts to evaluate a given attack tree $\mathcal{A}$ with respect to the total time required by the attacker to reach the root node. In a first step, the CDFs of the leaves are obtained as APH distributions from fitting algorithms [9, 10]. We then compress the APH distribution $[\![\mathcal{A}]\!]$ of $\mathcal{A}$ with the tool APHzip. For this purpose, we need to transform $\mathcal{A}$ into an expression which follows the grammar of APHzip. This can be done by traversing $\mathcal{A}$ in a depth-first manner: the labels of all nodes are listed in the order the nodes occur this traversal. Additionally, we have to take care of nested brackets and the fact that $\mathrm{con}(\cdot)$, $\max(\cdot)$ and $\min(\cdot)$ are binary operators in APHzip (for efficiency reasons).

Example 2. Consider the AT in Fig. 3. Starting at the root, we order the nodes according to their occurrence in a depth-first manner and put the corresponding *APH* operations in place, keeping track of nested brackets. This yields

$$\mathrm{con}(\exp(1), \min(\max(\exp(2), \exp(1)), \mathrm{erl}(2, 3)), \exp(5)).$$

As $\mathrm{con}(\cdot)$ above has three parameters, we need to nest another $\mathrm{con}(\cdot)$ expression with the latter two parameters to obtain a valid APHzip expression.

The CDF of such a compressed APH distribution can then be derived numerically on the basis of Equation (1): Given a set of time-points T and a PH representation $(\boldsymbol{\alpha}, \mathbf{A})$, we compute the probabilities $F(t)$ that the absorbing state is hit within t time-units, for each $t \in T$. To calculate these probabilities, we have implemented a postprocessing tool which uses the uniformization technique [32] and the Fox-Glynn algorithm [33]. Alternatively, several stochastic model checkers such as PRISM [34] or MRMC [35] can be employed as postprocessing tools.

The time-dependent analysis is a powerful tool since it basically performs a static analysis for any point in time. As an example, we fix the time horizon $t = 1$ and calculate the probability to reach the root of the model in Fig.3 with a static bottom-up evaluation. As there is no correspondence for the SEQ gate in static

models, we treat it as an AND gate. At first, we define random variable X_1 which corresponds to the BAS 'break into bank' and is distributed according to exp(1), so $X_1 \sim \exp(1)$. Similarly, define $X_2 \sim \exp(2)$, $X_3 \sim \exp(5)$ and $X_4 \sim \mathrm{erl}(2,3)$ to represent the other BASs. We have $\mathbb{P}(X_1 \leq 1) = 0.63$, $\mathbb{P}(X_2 \leq 1) = 0.87$, $\mathbb{P}(X_1 \leq 3) = 0.99$ and $\mathbb{P}(X_4 \leq 1) = 0.8$. In a static evaluation, the probability to reach an AND gate is the product of the probabilities of its inputs. An OR gate with input probabilities c_1 and c_2 is reached with probability $1-(1-v_1)(1-v_2)$. The probability to steal the gold in this static interpretation is thus calculated by

$$\mathbb{P}(\text{steal gold}) = 0.63 \cdot (1 - (1 - 0.87 \cdot 0.63) \cdot (1 - 0.8)) \cdot 0.99 = 0.57.$$

Evaluating the random variable C which is distributed according to the APH distribution in Example 2 after one time unit yields the same result with our tool chain: $\mathbb{P}(C \leq 1) = 0.57$.

Conservative Extension. The above example illustrates that time-dependent analysis conservatively extends the conventional static and thus untimed interpretation. To make this precise, we restrict to attack trees without SEQ gates (which are dynamic in nature), and relate to the conventional static probabilistic semantics [14].

We assume that each BAS vertex v has a probability p_v associated to it. Then the conventional static probabilistic semantics for a BAS is $\llbracket v \rrbracket = p_v$, for an AND gate it is $\llbracket \mathrm{AND}(v_1, \ldots, v_n) \rrbracket = \prod_1^n \llbracket v_i \rrbracket$, and for an OR gate it is $\llbracket \mathrm{OR}(v_1, \ldots, v_n) \rrbracket = 1 - \prod_1^n (1 - \llbracket v_i \rrbracket)$. This induces a static probabilistic semantics $\llbracket \mathcal{A} \rrbracket$ of any attack tree $\mathcal{A}$ (without SEQ gates), provided each BAS has a probability associated to it. It gives the probability to succesfully carry out the attack represented by the tree.

For a given attack tree $\mathcal{A}$ and time $t \in \mathbb{R}^+$, we now use $\mathcal{A}_t$ to refer to the attack tree where each BAS v gets the static probability $p_v = F_v(t)$ associated, where $F_v = L(v)$ is the CDF labelling vertex v. So we look at each BAS at time t and ask for the probability that the basic attack step represented by it is already completed. On the other hand, we can also take the timed interpretation $[\![\mathcal{A}]\!]$ of the entire tree, which by the semantics in Section 3 is some CDF $F_{\mathcal{A}}$ and look at the probability of successfully having completed the attack by time t by evaluating $F_{\mathcal{A}}(t)$. We refer to this value as $[\![\mathcal{A}]\!]_t$ in the theorem below.

Theorem 2. *Let $\mathcal{A}$ be an attack tree without SEQ gates. For any time point $t \in \mathbb{R}^+$, $[\![\mathcal{A}]\!]_t = \llbracket \mathcal{A}_t \rrbracket$.*

Thus, the time-dependent analysis conservatively extends static attack tree modelling. The proof uses the semantic interpretation of AND and OR gates as maximum and minimum operations, respectively (Section 3), the definition of the stochastic operations maximum and minimum (Section 2) and their properties, and the fact that as long as the underlying model has a tree structure, all nodes on the same level are stochastically independent.

5 Implications for System Security

Our analysis technique describes a system's vulnerabilities in a temporal context, but the results still need to be interpreted in terms of security. Generally, the analysis provides the user with a temporal dimension for possible attack vectors and associated risks expressed by success probabilities—can attacks be successfully executed in hours, days or rather months? More specific findings for security practitioners can be generated by elaborating on the primary results.

Lower Bound Analysis. In some cases the attacker has only a little time frame to execute an attack. An example for such a time-dependent vulnerability is the roll-out or update of an operating system. Once it is on the market, many hackers will try to find and exploit unknown vulnerabilities before they are found and fixed by the developers. In this race, the starting point of an attack is quite predictable. Thus, the analysis of $\mathbb{P}(0 < X < u)$ is of interest to answer the question 'How much time does the attacker need to succeed with a certain percentage'. The result gives an idea about how fast one has to react to fix vulnerabilities.

Cost-Benefit Analysis of Countermeasures. Security officers often face the problem to determine the benefits of investments into the existing security landscape. If more than one countermeasure might be beneficial but the budget is limited, a cost-benefit analysis is the classical tool to arrive at a decision. Our analysis framework helps to argue from a temporal perspective: 'Given countermeasures A, B and C, which one is most effective in reducing the risk of a successful attack within 1 day?'. This question can be answered by performing a separate analysis for each countermeasure option. We adapt the original attack tree by removing all subtrees and leaves that are prevented by the respective countermeasure and analyze the resulting model. The results can then be compared with respect to the desired properties.

Impact of Individual BASs. Naturally, one wants to identify the most serious vulnerabilities in a system. Therefore, a typical question which follows from the formation of an attack tree is the following: 'Which BAS has the most impact in the attack tree?'. In the context of our temporal analysis this question reads as 'The execution time of which BAS impacts the total execution time of the whole attack scenario most significantly?'. We can answer this by performing a sensitivity analysis. We perform $k + 1$ experiments, where k is the number of BASs. Initially, we analyse the original attack tree, and in each subsequent analysis we change the input distribution of one BAS, leaving the others fixed; *i.e.*, we adapt the parameters of the distribution of the considered BAS in such a way that the expected execution time of this BAS doubles. We can then determine in how far this change reflects in the execution time of the whole attack scenario. By comparing the results we can rank the BASs with respect to their impact on the total execution time of a successful attack. Moreover one can answer the question 'If the system is to be protected for one month, against which BAS it is

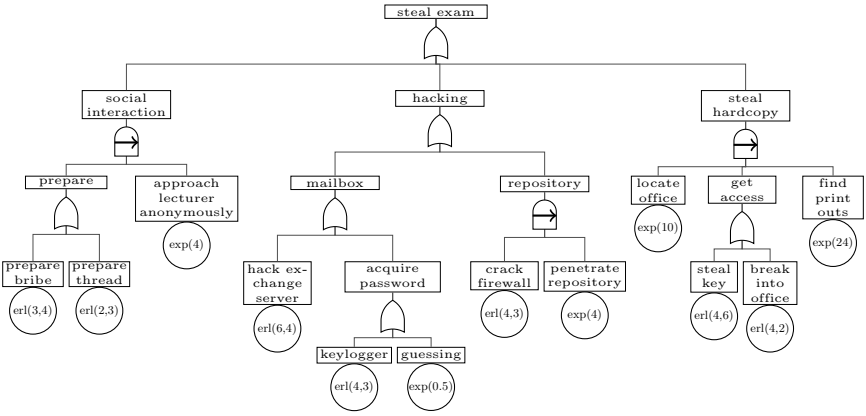
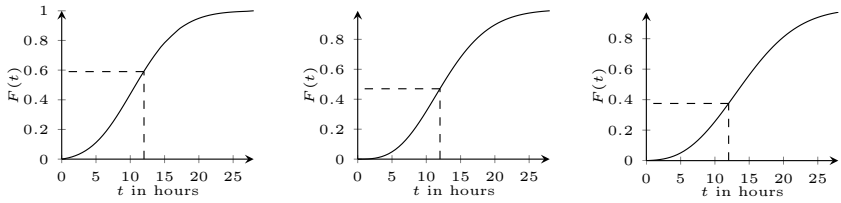


Fig. 5. Attack tree for the case study 'Steal exam'



(a) No countermeasures (b) Countermeasure A (c) Countermeasure B

Fig. 6. CDFs of the time to steal the exam with respect to different countermeasures

most important to protect?'. On the downside, this kind of sensitivity analysis requires $k + 1$ times more resources.

6 Case Studies

In this section we highlight the effectiveness of our approach by means of several case studies. They demonstrate that the algorithm implemented in APHzip yields significant state space compressions so that even complex scenarios can be analyzed efficiently, as presented in Table 2.

Steal Exam. This case study models a student who wants to get hold of a forthcoming exam. Within this scenario we consider three different types of attacks: social engineering, hacking and physical intrusion. Each attack type consists of various possible attack paths. To obtain a digital version of the exam via hacking, the student can, for instance, try to find a copy of the exam on either the mailbox server or the repository. Both possible attack paths can be refined in several BASs. For instance, the student could either hack the exchange server externally

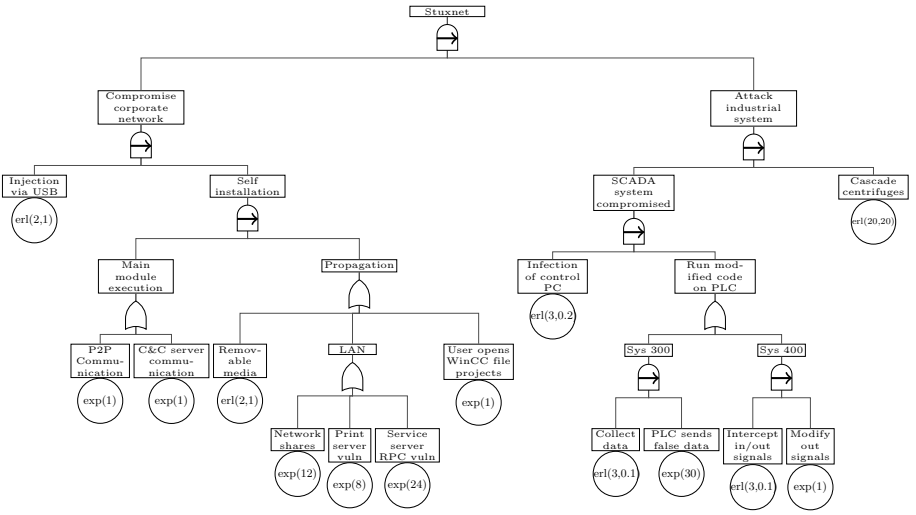


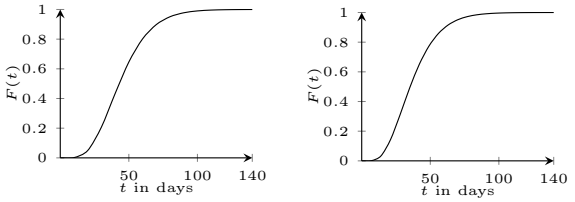
Fig. 7. Attack tree for the case study 'Stuxnet'

or try to acquire the account data by either using a keylogger or simply guessing the password. To each BAS we assign an exponential or an Erlang-distibution as estimates of the time required by the attacker to successfully execute this step.

The school wants to decrease the risk of such a theft, since it would heavily damage its reputation. The school management has the options to either acquire safe deposit boxes (countermeasure A) in which exams can be stored, or to apply a policy which forbids to send exams via email (countermeasure B). The first option would prevent the BAS 'steal key', since the lockers are password-protected. The second option would prevent the subtree 'mailbox'. We assume that the countermeasures block the corresponding BAS in their entirety. The school management wants to find the option which minimizes the risk of an attack within the opening time of the school (12 hours). We evaluated the attack scenario in the original set-up, and with respect to the two countermeasures. The results are displayed in Fig. 6. The attack tree without countermeasures applied is presented in Fig. 5.

The results clearly show that countermeasure B is more effective than countermeasure A. It reduces the risk of a successful attack within the given time by a third. The introduction of a policy is in this case more effective than an investment into the physical infrastructure. Of course, the application of both countermeasures would reduce the probability of a successful theft even more. This example highlights the strength of a timed analysis. It allows the security practitioner to evaluate the system security with respect to crucial time intervals.

Stuxnet. The second case study considers the Stuxnet attack, a sophisticated cyber attack which targeted industrial installations in 2010 and arouse great interest in media and among security experts. The model is based on [36]. To



(a) Original set-up (b) Increased rate of 'Infection of control PC'

Fig. 8. CDFs of the time to successfully execute the Stuxnet attack

Table 1. Sensitivity analysis of the 'Stuxnet' case study. The table shows the increase in the probability of success of the whole attack after 20, 40 and 60 days, if the rate of the respective step is doubled.

BAS	after 20 days	after 40 days	after 60 days
Injection via USB	+1.0 %	+2.3 %	+1.2 %
P2P Communication	+0.2 %	+0.3 %	+0.1 %
C&C server communication	+0.2 %	+0.3 %	+0.1 %
Removable media	+0.1 %	+0.2 %	+0.1 %
Infection of control PC	+7.4 %	+17.6 %	+17.6 %
Collect data	+2.5 %	+15.1 %	+16.3 %
Intercept in/out signals	+3.2 %	+12.3 %	+8.0 %
Modify out signals	+0.5 %	+0.7 %	+0.3 %
Cascade centrifuges	+0.8 %	+1.5 %	+0.7 %
other BASs	+0.0 %	+0.0 %	+0.0 %

adapt the case study to our formalism, we left out the probabilistic nodes used in this model.

The goal of Stuxnet in our scenario is to compromise supervisory, control and data acquisition systems (SCADA) of Iranian nuclear enrichment facilities to slow down the production of centrifugal machines. In the considered model, the business corporate network is assumed to be isolated from the SCADA system. Thus, the attack consists of the two phases; first the internal corporate network is compromised and then the SCADA system attacked from there.

As the corporate network is not directly connected to the Internet, an attack can be initiated by infecting an external device which is brought into the facilities and connected to the control system. Once it has installed itself on one PC, the malware tries to infect as many workstations as possible. It then waits until it can reach the Process Control Network, for instance through an infected removable drive. Within the SCADA system it can target modules with two specific CPU types. After gathering data for a longer period of time it sends faked data to the physical infrastructure which slows down and even damages the targeted centrifuges. The attack tree is presented in Fig. 8.

We want to determine the BAS which has the most impact upon the total execution of this attack and conduct a sensitivity analysis as explained above,

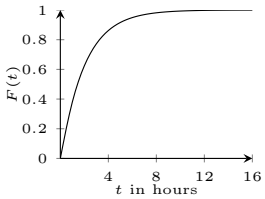


Fig. 9. CDF of the time to attack the IPTV system

i.e., we sequentially change the CDF of one BAS at a time such that its expected execution time doubles. The results are compared to the original model to identify the BASs which are most sensitive with respect to the total execution time. Table 1 highlights the results which suggest that the steps 'Infection of control PC', 'Intercept in/out signals' and 'Collect data' have the greatest impact upon the execution time of the whole attack and should be prioritized when it comes to the application of countermeasures.

IPTV case study. This case study was conducted in the European TREsPASS project [37] to gain insights into the modeling and analysis of socio-technical attacks. It describes a home-payment system designed to support elderly people or people with disabilities, who may have difficulty in leaving their home, in managing their own money. This system is based on the delivery of payments services to individuals via an IPTV set-top box in their own home. It allows the user to order and pay for food and goods of the daily life from home. The system is currently in the development phase and its potential attack vectors are investigated with the help of the following scenario: Since the system is specifically designed for elderly people, we assume that the attacker is a carer who intends to abuse the set-top box to enrich himself. He can for instance try to acquire the payment card and the password of the set-top box, either by stealing it or using his social engineering skills. In total, this case study covers eight major scenarios.

This case study is much more complex than the previous ones, the attack tree contains 148 nodes which are located on up to 14 different levels. More than 90 of its nodes are leaves. Due to confidentiality reasons, we cannot disclose more details about the tree.

The state space of the resulting APH model before compression has a size of about 10^{10}. To evaluate this model in a feasible time we used a shortcut that accelerates the analysis process by orders of magnitudes. Instead of composing the whole APH representation at once, we analyze the model in a compositional manner. At first, we split the attack tree into 9 subtrees which are connected via OR gates. We analyse these subtrees individually by computing the probability of the attacker's success for a set of time-points T. The results are recomposed by using the static computation formula for the OR gate for each $t \in T$. This approach is possible because the subtrees do not share nodes and are thus stochastically independent. This compositional analysis allowed us to evaluate the case

Table 2. Set-up and runtime performance of the case studies

	#Leaves	#Gates	#States before APHzip	#States after APHzip	Runtime in seconds
Steal Exam	12	9	15121	160	9.37
Steal Exam A	12	9	3025	76	1.26
Steal Exam B	12	9	5041	157	5.32
Stuxnet	14	11	94	56	1.13
Stuxnet Sensitivity Analysis	14	11	94	56	16.65
IPTV	92	56	647*	482*	551.62*

* sum over all 9 models of the compositional analysis

study within minutes, whereas the compression and analysis of the entire model at once would take several weeks. The result is presented in Fig.9.

Case Study Evaluation. Each attack tree in the case studies presented has a distinctive structure with leaves on different levels being connected by a mixture of gates. We analyzed each set-up with the APHzip toolchain. The results, as presented in Table 2, were computed with the webservice which implements APHzip, the CDFs were derived with the postprocessing tool on a meachine with a 2.20 GHZ dual-core processor. #States gives the number of states in the graphical representations of the APH of $[\![r]\!]$. The runtime is calculated as the sum of the runtime of both tools.

The smaller models in the first case study can be solved within seconds and even a more involved sensitivity analysis can be performed quickly. Unfortunately, previous work on time-dependent attacks does not offer any figures for comparison. Nonetheless, the results suggest that our analysis technique is extremely efficient in deriving a probability distribution for the execution time of an attack scenario expressed by an attack tree. With the IPTV attack scenario also a more complex case study could be solved in a feasible time.

Another interesting observation is that the complexity of the considered attack tree is not the most important factor when it comes to the computation time. Complex input distributions, such as Erlang distributions with many phases, cause a blow-up of the state space since they are rooted at the lowest level of the composition; and thereby cause longer computation times. Therefore, the computation time of the analysis of a model depends on both its scale as well as the complexity of its input distributions.

7 Conclusion

In this paper we have formalized the semantics of attack trees so as to allow their use for a probabilistic timed evaluation of attack scenarios. The semantics used an effective framework based on acyclic phase-type distributions, and as

such enables the derivation of the distribution of the time until the attack is successfully executed. We highlighted that any input distribution at hand for a basic attack step can be cast into the class of APH distributions. Its impact is propagated along the tree, yielding a single monolithic APH distribution for the entire tree. A key component for effective and efficient evaluation is a compression algorithm for APH distributions. This compression can be weaved into the construction process of the monolithic distribution, a feature that is especially interesting in light of the dynamic nature of attack trees: it allows the preprocessing of often appearing attack paths.

We have reported on several distinct case studies demonstrating that this approach can evaluate complex scenarios in a short time. These empirical studies have been carried out with an implementation of the approach as part of a tool chain. As future work we aim at merging and enriching the existing tool with a state-of-the-art APH fitting algorithm [10] as well as a graphical interface. We further aim at support for cost annotations of attack steps, and at support for probabilistic gates, as they appear in the literature on attack trees [21, 22].

Acknowledgements. This research has been partially funded by the NWO project ArRangeer (12238), by the DFG/NWO bilateral project ROCKS (DN 63-257), by the DFG SFB/TR 14 (AVACS), and by the EU FP7 projects no. 295261 (MEALS), 318003 (TREsPASS), and 318490 (SENSATION).

References

1. Basin, D.A., Capkun, S.: The research value of publishing attacks. Commun. ACM 55(11), 22–24 (2012)
2. Köpf, B., Malacaria, P., Palamidessi, C.: Quantitative Security Analysis (Dagstuhl Seminar 12481). Dagstuhl Reports 2(11), 135–154 (2013)
3. Schneier, B.: Attack trees: Modeling security threats. Dr. Dobb's Journal 24(12) (December 1999)
4. Jürgenson, A., Willemson, J.: Computing exact outcomes of multi-parameter attack trees. In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part II. LNCS, vol. 5332, pp. 1036–1051. Springer, Heidelberg (2008)
5. Kordy, B., Pouly, M., Schweitzer, P.: Computational aspects of attack–defense trees. In: Bouvry, P., Kłopotek, M.A., Leprévost, F., Marciniak, M., Mykowiecka, A., Rybiński, H. (eds.) SIIS 2011. LNCS, vol. 7053, pp. 103–116. Springer, Heidelberg (2012)
6. Roy, A., Kim, D., Trivedi, K.: Attack countermeasure trees (act): towards unifying the constructs of attack and defense trees. Sec. and Commun. Netw. 5(8), 929–943 (2012)
7. Zonouz, S., Khurana, H., Sanders, W., Yardley, T.: Rre: A game-theoretic intrusion response and recovery engine. In: IEEE/IFIP International Conference on Dependable Systems Networks, DSN 2009, pp. 439–448 (July 2009)
8. Ray, I., Poolsapassit, N.: Using attack trees to identify malicious attacks from authorized insiders. In: de Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 231–246. Springer, Heidelberg (2005)

9. Horváth, A., Telek, M.: PhFit: A general phase-type fitting tool. In: Field, T., Harrison, P.G., Bradley, J., Harder, U. (eds.) TOOLS 2002. LNCS, vol. 2324, pp. 82–91. Springer, Heidelberg (2002)
10. Thümmler, A., Buchholz, P., Telek, M.: A novel approach for phase-type fitting with the EM algorithm. IEEE Trans. Dependable Sec. Comput. 3(3), 245–258 (2006)
11. Weiss, J.: A system security engineering process. In: Proceedings of the 14th National Computer Security Conference, pp. 572–581 (1991)
12. Amoroso, E.: Fundamentals of computer security technology. Prentice-Hall, Inc., Upper Saddle River (1994)
13. Kordy, B., Pietre-Cambacedes, L., Schweitzer, P.: DAG-based attack and defense modeling: Don't miss the forest for the attack trees. CoRR abs/1303.7397 (2013)
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)
15. Bistarelli, S., Peretti, P., Trubitsyna, I.: Analyzing security scenarios using defence trees and answer set programming. Electron. Notes Theor. Comput. Sci. 197(2), 121–129 (2008)
16. 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)
17. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.: Automated generation and analysis of attack graphs. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 273–284 (2002)
18. McQueen, M., Boyer, W., Flynn, M., Beitel, G.: Quantitative cyber risk reduction estimation methodology for a small SCADA control system. In: Proceedings of the 39th Annual Hawaii International Conference on System Sciences, HICSS 2006, vol. 9, pp. 226 (2006)
19. LeMay, E., Ford, M.D., Keefe, K., Sanders, W.H., Muehrcke, C.: Model-based security metrics using adversary view security evaluation (advise). In: Proceedings of the 2011 Eighth International Conference on Quantitative Evaluation of SysTems, QEST 2011, pp. 191–200. IEEE Computer Society (2011)
20. McDermott, J.: Attack net penetration testing. In: Proceedings of the 2000 Workshop on New Security Paradigms, NSPW 2000, pp. 15–21. ACM, New York (2000)
21. Piètre-Cambacédès, L., Bouissou, M.: Beyond attack trees: Dynamic security modeling with boolean logic driven Markov processes (BDMP). In: European Dependable Computing Conference (EDCC), pp. 199–208 (April 2010)
22. Piètre-Cambacédès, L., Bouissou, M.: Attack and defense modeling with BDMP. In: Kotenko, I., Skormin, V. (eds.) MMM-ACNS 2010. LNCS, vol. 6258, pp. 86–101. Springer, Heidelberg (2010)
23. Johnson, M.A., Taaffe, M.R.: The denseness of phase distributions. School of Industrial Engineering Research Memoranda 88-20, Purdue University (1988)
24. Asmussen, S., Nerman, O., Olsson, M.: Fitting phase-type distributions via the EM algorithm. Scandinavian Journal of Statistics 23(4), 419–441 (1996)
25. Neuts, M.F.: Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach. Dover (1981)
26. He, Q.M., Zhang, H.: Spectral polynomial algorithms for computing bi-diagonal representations for phase type distributions and matrix-exponential distributions. Stochastic Models 2(2), 289–317 (2006)
27. Cox, D.R.: A use of complex probabilities in the theory of stochastic processes. Proceedings of the Cambridge Philosophical Society 51(2), 313–319 (1955)

28. Cumani, A.: Canonical representation of homogeneous Markov processes modelling failure time distributions. Microelectronics and Reliability 2(3), 583–602 (1982)
29. Pulungan, R., Hermanns, H.: Acyclic minimality by construction—almost. In: QEST, pp. 63–72. IEEE Computer Society (2009)
30. Buchholz, P.: Exact and ordinary lumpability in finite Markov chains. Journal of Applied Probability 31, 59–75 (1994)
31. Jonsson, E., Olovsson, T.: A quantitative model of the security intrusion process based on attacker behavior. IEEE Transactions on Software Engineering 23(4), 235–245 (1997)
32. Reibman, A.L., Trivedi, K.S.: Numerical transient analysis of Markov models. Computers & OR 15(1), 19–36 (1988)
33. Fox, B.L., Glynn, P.W.: Computing poisson probabilities. Commun. ACM 31(4), 440–445 (1988)
34. Kwiatkowska, M.Z., Norman, G., Parker, D.: Probabilistic symbolic model checking with prism: a hybrid approach. STTT 6(2), 128–142 (2004)
35. Katoen, J.P., Zapreev, I.S., Hahn, E.M., Hermanns, H., Jansen, D.N.: The ins and outs of the probabilistic model checker mrmc. Perform. Eval. 68(2), 90–104 (2011)
36. Kriaa, S., Bouissou, M., Pièvre-Cambacédès, L.: Modeling the stuxnet attack with BDMP: Towards more formal risk assessments. In: 7th International Conference on Risk and Security of Internet and Systems (CRiSIS), pp. 1–8 (October 2012)
37. The TREsPASS project: `http://www.trespass-project.eu`