# State-Based Analysis in ADVISE

Michael D. Ford* Peter Buchholz† William H. Sanders*

*Dept. of Electrical and Computer Engineering,
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1308 W. Main St., Urbana, IL, U.S.A.
{mdford2, whs}@illinois.edu

†Informatik IV, TU Dortmund
D-44221 Dortmund, Germany
peter.buchholz@cs.tu-dortmund.de

***Abstract*—There is an increasing need for quantitative security metrics to empower the decision-making of system architects and administrators. We previously defined the ADVISE method, which combines a state-based security model and an adversary profile to generate quantitative metrics through simulation. Since simulation is often costly, particularly when applied to the analysis of rare events like successful attacks against a system, we extend the analysis methods of ADVISE to Markov chain analysis techniques that enable the efficient and accurate computation of a wide variety of quantitative security measures. With these analysis approaches, it is possible to extend the type of metrics available when using the ADVISE method while lifting previous restrictions placed on the adversary profile.**

***Keywords*-security, quantitative analysis, numerical methods**

## I. Introduction

In real system architectures, security can often be seen as a quantitative measure, since the trade-off between the cost of making a system more secure and the cost and probability of a successful attack must be considered. Quantitative security measures help to identify the steps that will be most effective in making a system more secure. For those reasons, there is an increasing need for quantitative security analysis of systems. Several styles of security analysis can be performed on a system. While some approaches only focus on analysis of live systems, which means that they perform some form of monitoring [8], others use model-based techniques to evaluate the security of a system [7]. The ADVISE method [3], [4] enables preproduction system analysis in addition to the analysis of live systems.

In our previous work, we have demonstrated the usefulness of the ADVISE approach of marrying state-based security models with adversary profiles to provide predictive metrics about how specific adversaries will attack a network. In this paper, we extend that work to enable metric generation through the use of state-based methods that result in Markov processes that can be analyzed with numerical methods in addition to simulation. In this paper we first give an overview of ADVISE and provide definitions used throughout the remainder of the paper. Next, we present algorithms for the creation of the state space and the exploration of the transition matrix. Then, we describe the new metrics that are enabled through this translation and propose numerical methods to analyze the different metrics. Finally, we consider implementation issues, an example, and future work and give concluding remarks.

## II. ADVISE and Basic Definitions

The ADversary VIew Security Evaluation (ADVISE) method produces quantitative security metrics by combining a system model and an adversary profile into an executable model. The system is modeled as an attack execution graph (AEG), which consists of a set of attack steps that an adversary may choose to attempt. This type of state-based model has a long history in the security metric literature, beginning with [8].

Lippmann et al. present an annotated review of attack graph work prior to 2005 in [5], and more recent work is summarized in [10]. Both reviews focus largely on automatic attack graph generation, and praise easily computable metrics, such as reachability analysis and perimeter hardening. In contrast to other approaches, ADVISE does not limit the scope of the system model or the set of metrics that can be computed. ADVISE incorporates timing, cost, preconditions, probabilistic outcomes, and detection probabilities of attack steps in its system description in order to enable description of a wide variety of complex metrics.

When attacking a network, an adversary usually follows some plan of attack and makes his or her decisions based on some rational planning that incorporates cost functions evaluating the benefits of different attack steps. ADVISE incorporates these adversary preferences through an adversary profile (AP). Adversary profiling has been used to compare the attractiveness of entire attack sequences in [1] and [12]. ADVISE assesses the relative attractiveness of individual attack steps to the adversary. Thus, the execution of an ADVISE model mimics the optimization of a Markov decision process [9] or an even more complex stepwise decision process; the adversary selects an attack step to attempt, but the outcome, and therefore the resulting next state, is selected probabilistically. For a given adversary, ADVISE chooses the attack steps deterministically based on the state, by selecting the attack steps that maximize the profit according to possible outcomes and related probabilities. The planning horizon denotes the number of steps an adversary may look into the future while selecting an attack step. In that way, selection of the next attack steps corresponds to the computation of an optimal policy in a decision process. By precomputing the attack step decision and removing all nonoptimal decisions, we reduce the problem to one that can be analyzed using state-space metrics. Thus, for analysis we consider only the optimal decision of an adversary and not the huge set of all possible decisions.

### A. *ADVISE Notation*

The following description recapitulates the ADVISE model and the corresponding notations. It is taken from [3], [4] and is only slightly modified for the purpose of state-based analysis.

The goal of this state-based analysis is the computation of security-related quantitative measures with respect to one adversary attempting to attack a system. Let $X$ be the unknown state space according to a predefined attack execution graph and $s_0 \in X$ be the known initial state. A state is defined as a vector of Boolean values ranging over the sets $R$, $K$, and $G$. $R$ is the set of *access domains* within the system that the adversary may access, $K$ is the set of *knowledge items* the adversary possesses, and $G$ is the set of *attack goals* the adversary has achieved. An example of an access domain is the read permission for some directory that an adversary may or may not have; a knowledge item could be knowledge of the name of a file that includes important information; and an attack goal could be a requirement to obtain information from a file. The complete reachable state space $X$ results from sequences of attack steps that are performed by the adversary according to his or her decision procedure. Each attack step may modify the state. An algorithm for state-space generation is introduced below, after the formal definition of an attack step.

$A$ is the set of attack steps and $a_i \in A$ is defined as

$$a_i = \langle B_i, T_i, C_i, O_i, Pr_i, F_i, E_i, G_i \rangle.$$

Table I contains a description of the components of an attack step, where $s \in X$ is the state of the system.

Table I
ATTACK STEP COMPONENT NOTATION

| | | |
|---|---|---|
| $B_i$ | $X \rightarrow \{true, false\}$ | Attack step Boolean precondition. |
| $T_i$ | $X \rightarrow dist_{\geq 0}$ | Attack step time distribution. |
| $C_i$ | $X \rightarrow \mathbb{R}^{\geq 0}$ | Attack step cost. |
| $O_i$ | | Finite set of attack outcomes. |
| $Pr_i$ | $X \times O_i \rightarrow [0,1]$ | Outcome probability distribution. |
| $F_i$ | $X \times O_i \rightarrow [0,1]$ | Probability of nondetection for each outcome. |
| $E_i$ | $X \times O_i \rightarrow X$ | Outcome-dependent state transition function. |
| $G_i$ | $X \times O \rightarrow \mathbb{R}$ | Gain for the state transition $s \rightarrow E_i(s,o)$. |

We briefly summarize the meaning of some of the components and refer the reader to [3], [4] for further details. The precondition $B_i(s)$ indicates whether the adversary possesses the skill, access and knowledge to perform the attack step in the current situation defined by state $s$. $T_i(s)$ is a random variable describing the duration of the attack step. Its range, $dist_{\geq 0}$, is the set of distributions with non-negative support. An attack can have different (usually two) outcomes, which occur with a predefined probability. Thus, after performing an attack step, the successor state is chosen probabilistically. In addition to the original definition in [3], [4], we have added the components $G_i$, which describes the gain of an attack step with some outcome, and $F_i$, which equals the probability that the attack step remains undetected.

In a state $s \in X$, a unique attack step is chosen by the function $\beta : X \rightarrow A$, which depends on the adversary and is described below. We assume that $\beta(s)$ is a deterministic function, which implies that the adversary chooses deterministically his or her next step. Therefore, in state $s$, the index $i$ as given in Table I is replaced with $\beta(s)$. E.g., the sojourn time $T(s)$ in state $s$ equals $T_{\beta(s)}(s)$. The gain $G_{\beta(s)}$ is associated with transitions as well as states, since it depends on the outcome of the attack step and therefore on the successor state. There may be multiple transitions from a state to a successor state since multiple outcomes of a single attack step may result in the same transition. State $s$ has up to $|O_{\beta(s)}|$ successor states, and we write $s \xrightarrow{(o,g,p_f,p_t)} s'$ where $o \in O_{\beta(s)}$, $g = G_{\beta(s)}(s,o)$ is the gain of the attack step, $p_f = F_{\beta(s)}(s,o)$, $p_t = Pr_{\beta(s)}(s,o)$, and $s' = E_{\beta(s)}(s,o)$. Additionally, we use the notation $s \xrightarrow{i} s'$ and $s \xrightarrow{i,o} s'$ for a transition from $s$ to $s'$ via attack step $a_i$ with an arbitrary outcome and with an outcome of $o$, respectively. The determination of $a_i \in A$ in state $s \in X$ determines all possible transitions, but the opposite is not generally true.

The state space $X$ is determined according to an adversary profile,

$$AP = \langle s_0, L, V, w_C, w_P, w_D, U_C, U_P, U_D, N \rangle,$$

as defined in [3, pages 10-11] and [4]. $L$ and $V$ are the attack skill level and the attack goal level functions summarizing the skills of the adversary in performing different variants of attacks that are encoded in the set of attack steps; $w_C, w_P$, and $w_D$ are the attack preference weights for cost, payoff, and detection probability; and $U_C, U_P$, and $U_D$ are the utility functions for cost, payoff, and detection probability. Those functions are used to compute the *best attack step* to be performed next (i.e., the results of the function $\beta(s)$), as shown below. $N$ is the planning horizon that determines the number of steps an adversary is able to look into the future to determine the cost of an attack step.

The central goal is to determine the attack step $a_i$ that will be performed by an adversary $AP$ in state $s$, or, in other words, to determine the function $\beta(s)$ for a predefined $AP$. We assume that attack steps in $A$ are ordered such that if the evaluation function results in several attack steps with the same maximal gain, there is a deterministic order that can be used to choose one step. That order might, for example, be established by ordering attack steps in $A$ according to their mean time requirements or costs or any other metrics.

In [3], the computation of $\beta(s)$ is performed using dynamic programming. That is, an approximation is used that explores the state space based on the adversary's look-ahead, $N$. Because of the nondeterministic nature of an attack step's outcome, the exploration results in the creation of a tree of successor states. Then, an attack step is selected at each level, starting with the leaves, and its attractiveness is propagated back up the tree. The process is repeated recursively, and an optimal first attack step can be chosen.

Of particular note is the fact that the method always selects a "best" attack step at each level. This is equivalent to assuming that the attractiveness calculation is a Bellman equation [9], or that the process is a Markov Decision Process. If the utility functions used in the attractiveness calculation are not of a linear form for payoff and cost, and of an exponential form for detection, then the computation might not select an optimal attack step, since the Bellman equations do not hold in this case. This will be shown in the next section together with an extended form of the computation of optimal decisions.

## B. Generalizing the Computation of Optimal Attack Steps

We now consider the computation of $\beta(s)$ in a more general setting than in [3], [4], going beyond the stepwise decision-making done in Markov decision processes. We use $h^t$ to denote a history with $t$ time steps.

$$h^t = \langle s_0, a_0, o_0, s_1, \ldots, s_{t-1}, a_{t-1}, o_{t-1}, s_t\rangle$$

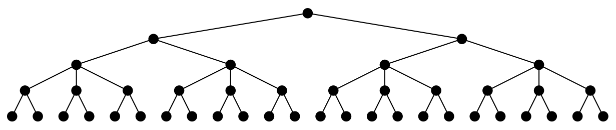
A history consists of a sequence of states $s_0, \ldots, s_t \in X$ such that $s \overset{a_i,o_i}{\longrightarrow} s_{i+1}$ for $0 \le i < t$. We use the notation $H^t(\langle s_0\rangle)$ to denote the set of all possible histories $h^t$, with look-ahead $t$, that begin with state $s_0$, and we extend this definition recursively as follows:

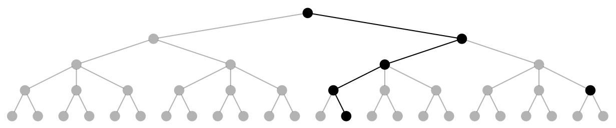
$$H^t(H^s) = \begin{cases} H^s & t = s \\ \{h^{t-1} \oplus \langle a, o, E_a(s_{t-1}, o)\rangle \mid \\ \quad h^{t-1} = (s_0, \ldots, s_{t-1}) \in H^{t-1}(H^s), \\ \quad a \in A : B_a(s_{t-1}) = \mathit{true}, o \in O_a\} & t > s \end{cases}$$

where the symbol $\oplus$ represents the append operator that appends a new step, outcome, and state to a history, i.e., $\langle s_0, a_0, o_0, s_1, \ldots, s_{t-1}, a_{t-1}, o_{t-1}, s_t\rangle \oplus \langle a_t, o_t, s_{t+1}\rangle = \langle s_0, a_0, o_0, \ldots, a_t, o_t, s_{t+1}\rangle$.

Figure 1(a) depicts the set of all possible histories $H^t$ within the state space of depth $t$. That is, from the initial state, the adversary selects one of two attack steps, each of which will result in either of two possible outcomes. From those resulting states, the adversary may select from three attack steps, again with two possible outcomes each. Figure 1(b) depicts a single history $h^t$ within the set of all possible histories extending from the initial state.

Notice that $h^t$ is a single path through the state space, while $H^t$ represents the entire set of paths through the state space of depth $t$. A history $h^t$ is the path an adversary could take if both attack steps and outcomes were fixed. $H^t$ is the set of paths an adversary could take if neither the attack steps nor outcomes were fixed. What remains is to depict the potential paths of an adversary given the deterministic choice of the *best attack step* with respect to a state.


(a) $H^t$: The Set of All Possible Paths


(b) $h^t$: A Single Path

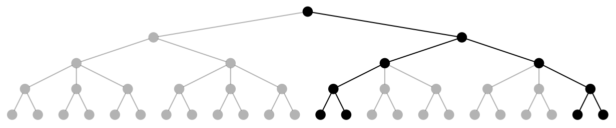

(c) $H^t_\sigma$: The Set of Possible Paths Given a Policy

Figure 1. History Trees

Therefore, we define a policy $\sigma$ to be a mapping at each look-ahead level from a state to an attack and $\sigma^t$ to be a policy with a look-ahead of $t$. Then $\sigma^t = (f_1, \ldots, f_t)$, where $f_k : X \to A$, i.e., $f_k(s) = a$ means that attack $a$ is chosen in step $k$. For $\sigma^t = (f_1, \ldots, f_t)$ and $\sigma'^u = (f'_1, \ldots, f'_u)$, we define $\sigma^t\sigma'^u$ as a policy with look-ahead $t + u$ that results from the concatenation of $\sigma^t$ and $\sigma'^u$ such that $\sigma^t\sigma'^u = (f_1, \ldots, f_t, f'_1, \ldots, f'_u)$. The set of all policies with look-ahead $t$ is $\Omega^t$, and $\Omega^t_i$ is the set of policies with look-ahead $t$ and attack step $a_i$ as the first attack step. Set $\Omega^t_i$ is empty in state $s$ if $B_i(s) = \mathit{false}$. A policy-consistent history is a history $h^t_\sigma$ such that $\forall k,\ f_k(s_k) = a_k$. Then, the set of all policy-consistent histories, denoted by $H^t_\sigma(\langle s\rangle)$, is the set of all possible sample paths through the state space starting with state $s$ given that the adversary follows $\sigma$ and only the outcomes are stochastic. See Figure 1(c).

With those notations, we can extend the notation for reward values that are required for the computation of the attractiveness.

$$CC^n(H^n(\langle s\rangle)) = \sum_{h^n \in H^n}\left(\sum_{i=0}^{n-1} C_i(s_i) \prod_{i=0}^{n-1} Pr_i(s_i, o_i)\right)$$

$$PP^n(H^n(\langle s\rangle)) = \sum_{h^n \in H^n}\left(\sum_{i=0}^{n-1} G_i(s_i, o_i) \prod_{i=0}^{n-1} Pr_i(s_i, o_i)\right)$$

$$DD^n(H^n(\langle s\rangle)) = \sum_{h^n \in H^n}\left(\prod_{i=0}^{n-1} F_i(s_i, o_i) \prod_{i=0}^{n-1} Pr_i(s_i, o_i)\right)$$

The values include the cumulative cost ($CC^n$), gain ($PP^n$), and probability that the attack is not detected

($DD^n$) over all possible histories of length $n$. With those values, we define the attractiveness of an attack policy $\sigma$.

$$\begin{aligned} at(H_\sigma^n(\langle s \rangle)) = \; & w_C U_C(CC^n(H_\sigma^n(\langle s \rangle))) \\ & + w_P U_P(PP^n(H_\sigma^n(\langle s \rangle))) \\ & + w_D U_D(DD^n(H_\sigma^n(\langle s \rangle))) \end{aligned} \tag{1}$$

$at()$ is the sum of weighted utilities of the cumulative cost, gain, and probability of nondetection that results from a specific policy. Observe that the functions $U_{(.)}$ in (1) can be arbitrary nonlinear functions. We now define the attractiveness of an attack step $a_i$.

$$attr^n(s, a_i) = \max_{\sigma \in \Omega_i^n} \left( at(H_\sigma^n(\langle s \rangle)) \right) \tag{2}$$

Then, $attr^n()$ is the maximum possible weighted utility for all policies that begin with the attack step $a_i$. Function $\beta$ is defined for a look-ahead of $N$ as

$$\beta(s) = \operatorname*{argmax}_{a_i \in A} \left( attr^N(s, a_i) \right). \tag{3}$$

In general, as already noted in [3, Theorem 3.2], the computation of $\beta(s)$ requires the computation of $at(s, .)$ for all policies of length $N$, which means that for each of the $(S \cdot A)$ policies, $|A|^{|O|^{N-1}}$ paths must be considered. That becomes impractical for larger values of $N$. Consequently, we now consider cases where the computation over all paths can be avoided. Of course, that implicitly or explicitly puts some restrictions on the functions $U_{(.)}$.

### C. An Optimality Ordering

In order to limit computation, we utilize a partial ordering or a total ordering that results in paths being eliminated from the attractiveness calculation. First, observe that the attractiveness of using the policy $\sigma$ in state $s$ is characterized by three values: $c = CC^n(H_\sigma^n(\langle s \rangle))$, $p = PP^n(H_\sigma^n(\langle s \rangle))$, and $d = DD^n(H_\sigma^n(\langle s \rangle))$. We define an order $\gg$ on the triples $(c, p, d)$.

To formally define $(c, p, d) \gg (c', p', d')$, let $\sigma_1^t, \sigma_2^t \in \Omega^t$ such that $c = CC^n(H_{\sigma_1^t}^n(\langle s \rangle))$, $p = PP^n(H_{\sigma_1^t}^n(\langle s \rangle))$, $d = DD^n(H_{\sigma_1^t}^n(\langle s \rangle))$, and $c' = CC^n(H_{\sigma_2^t}^n(\langle s \rangle))$, $p' = PP^n(H_{\sigma_2^t}^n(\langle s \rangle))$, $d' = DD^n(H_{\sigma_2^t}^n(\langle s \rangle))$. We say that $(p, d, c) \gg^n (p', g', c')$, $n > t$ if for all $l = 1, \ldots, n - t$, all $\sigma^l \in \Omega^l$ and all $s \in X$, $at(H_{\sigma^l \sigma_1^t}^{l+t}(s)) \geq at(H_{\sigma^l \sigma_s^t}^{l+t}(s))$. We say $(c, p, d) \gg (c', p', d')$ if $(c, p, d) \gg^n (c', p', d')$ for all $n$.

That is, we consider two possible futures associated with two distinct policies, $\sigma_1^t$ and $\sigma_2^t$. If $\sigma_1^t$ outperforms $\sigma_2^t$ for the attractiveness calculations of all past histories, then we use the relation $\gg$.

Usually, the functions $U_{(.)}(.)$ are of such a form that $(c, p, d) \gg (c', p', d')$ holds if $c \leq c'$, $p \geq p'$, and $d \geq d'$, i.e., the costs are smaller and the gain and the probability of not being detected are higher. This implies that the utility functions are monotonically increasing. However, there may be other cases, depending on the definition of the utility functions.

There is a specific situation that we will call the *MDP-case* (see also [3]). For this case, $\beta^n(s)$ can be defined recursively. Then, $\beta^1(s) =$

$$\operatorname*{argmax}_{a_i \in A} (w_C U_C(C_i(s)) + w_P U_P(P_i(s)) + w_D U_D(F_i(s)))$$

with $C_*^1(s) = C_{\beta^1(s)}(s)$, $P_*^1(s) = P_{\beta^1(s)}(s)$, and $D_*^1(s) = D_{\beta^1(s)}(s)$. That makes it possible to compute recursively $C_*^n(s), P_*^n(s), D_*^n(s)$ as well as $\beta^n(s)$ (see [3, Eq. (3.8)-(3.12)]. Let $v = w_C U_C(c) + w_P U_P(p) + w_D U_D(d)$ and $v' = w_C U_C(c') + w_P U_P(p') + w_D U_D(d')$.

Then in the MDP-case, the relation $\gg$ is defined as follows:

$$v > v' \Leftrightarrow (c, p, d) \gg (c', p', d').$$

Furthermore, if the two values are identical, we define

$$v = v' \Leftrightarrow (c, p, d) \approx (c', p', d'),$$

and in this case, it does not matter whether we choose $(c, p, d)$ or $(c', p', d')$. Consequently, in the MDP-case $\gg$ and $\approx$ define a total order, i.e., two values $(c, p, d)$ and $(c', p', d')$ are in relation $\gg$ or in relation $\ll$ or in relation $\approx$. Below, we use the notation $(c, p, d) \underline{\gg} (c', p', d')$ for $(c, p, d) \gg (c', p', d')$ or $(c, p, d) \approx (c', p', d')$ and use this relation in the second algorithm of Section III-A for evaluating function $\beta(s)$.

## III. State Space and Matrix Generation

In this section, we present the algorithms associated with state-space generation, attack step selection, and transition matrix generation. Those components are critical to enabling the use of solution techniques on attack execution graphs and a corresponding adversary profile's counterparts.

### A. State-Space Generation

Let $\hat{X}$ be the potential state space that contains all combinations of values of the Boolean state variables and, consequently, has a cardinality of $2^{|R| \cdot |K| \cdot |G|}$. In ADVISE models, the potential state space may be too large to be enumerated; however, the reachable state space $X$ usually contains only a very small subset of the states.

For the basic state-space generation algorithm, we use a data structure $\mathcal{U}$ to store state descriptions and a data structure $\mathcal{X}$ to store state description plus $\beta(s)$. Then, Algorithm 1 is used for state-space generation.

**Algorithm 1** State-Space Generation

```
1: $\mathcal{U} = \{s_0\}$;
2: **while** $\mathcal{U} \neq \emptyset$ **do**
3:     remove $s$ from $\mathcal{U}$;
4:     $a_i$ = determine_beta($s$);
5:     **for all** $o \in O_i$ **do**
6:         $s' = E_i(s, o)$;
7:         **if** $s' \notin \mathcal{X} \cup \mathcal{U} \cup \{s\}$ **then**
8:             $\mathcal{U} = \mathcal{U} \cup \{s'\}$;
9:     $\mathcal{X} = \mathcal{X} \cup \{(s, a_i)\}$;
```

For finite state spaces, the algorithm terminates after computing the set of states and the index of the attack step to be performed in the state. That information is sufficient to characterize the complete stochastic process.

The crucial stage of state-space generation is the evaluation of the function *determine_beta()*. If no additional information is available, the function has to evaluate all strings from $\Omega^N$. It can do so using Algorithm 2, in which $\bullet$ denotes *undefined*.

**Algorithm 2** Simple $\beta(s)$ Computation

```
1:  $\beta = \bullet$ and $best\_val = -\infty$;
2:  for all $\sigma \in \Omega^N$ do
3:      determine $val = at(H^n_\sigma(\langle s \rangle))$ using (eq. 1);
4:      if $val > best\_val$ then
5:          $best\_val = val$;
6:          $\beta = f_1(s), \sigma = (f_1, \ldots)$;
7:  return $\beta$;
```

The procedure is rather naive and may compute values recursively several times. If the relation $\gg$ is available, a graph-based approach that can exploit it is much more efficient. In that case, successor states up to $N$ steps apart are explored, and an acyclic multigraph is generated. The algorithm for determining $\beta(s)$ consists of a forward and a backward phase. In the forward phase, all states are generated that can be reached in $1, 2, \ldots, N$ steps, and they are stored in sets $\mathcal{Y}_n$. Furthermore, all transitions $s \overset{(a_i,o)}{\rightarrow} s'$ with $s \in \mathcal{Y}_n$ and $s' \in \mathcal{Y}_{n+1}$ are stored with the source states to be used in the subsequent backward phase. Since several transitions can exist between two states $s$ and $s'$, the graph is a multigraph. Algorithm 3 describes the forward phase.

**Algorithm 3** Forward $\beta(s)$ Computation

```
1:  $\mathcal{Y}_0 = \{s_0\}$;
2:  for $n = 1$ to $N$ do
3:      $\mathcal{Y}_n = \emptyset$;
4:      for all $s \in \mathcal{Y}_{n-1}$ do
5:          for all $a_i \in A$ with $B_i(s) = true$ do
6:              for all $o \in O_i$ do
7:                  let $s' = E_i(s, o)$;
8:                  $\mathcal{Y}_n = \mathcal{Y}_n \cup \{s'\}$;
9:                  add an edge between $s$ and $s'$ labeled with
                    $(a_i, o)$ to state $s$;
10: return $\mathcal{Y}_0, \ldots, \mathcal{Y}_N$;
```

In the subsequent backward phase (see Algorithm 4), the values are evaluated bottom-up in the multigraph. Since the valuation is computed bottom-up, we store together with each state $s$ a set of valuations $s.\mathcal{V}$. A valuation is of the form $(c, p, d)$, with the same meaning described above. For the states $s \in \mathcal{Y}_N$, $s.\mathcal{V} = \{(0, 0, 1)\}$ is used for initialization; the other values are computed in Algorithm 4.

Step 5, in which all combinations of possible valuations in successor states have to be evaluated, is crucial. Since $O_i$ usually has two outcomes, success or failure, the number of possible valuations grows with the product of

**Algorithm 4** Backward $\beta(s)$ Computation

```
1:  $\beta = \bullet$ and $best\_val = -\infty$;
2:  for $n = N - 1$ downto $0$ do
3:      for all $s \in \mathcal{Y}_n$ do
4:          for all $a_i \in A$ with $B_i(s) = true$ do
5:              for all combinations $(c', p', d') \in (s', n+1).\mathcal{V}$
                where $s \overset{(a_i,o)}{\longrightarrow} s'$ do
6:                  $c = C_i(s) + \sum_{o \in O_i} Pr_i(s, o)c'$;
7:                  $p = \sum_{o \in O_i} Pr_i(s, o)(G_i(s, o, s') + g')$;
8:                  $d = \sum_{o \in O_i} Pr_i(s, o)(F_i(s, o)d')$;
9:                  if $n = 0$ then
10:                     $val = w_C U_C(c) + w_P U_P(p) + w_F U_F(d)$;
11:                     if $val > best\_val$ then
12:                         $best\_val = val$;
13:                         $\beta = a_i$;
14:                 else if $\nexists\ (c, p, d)' \in s.\mathcal{V} \mid (c, p, d)' \gg (c, p, d)$
                    then
15:                     $(s, n).\mathcal{V} = (s, n).\mathcal{V} \cup \{(c, p, d)\}$;
16:                     remove all $(c, p, d)'$ from $(s, n).\mathcal{V}$ with
                        $(c, p, d) \gg (c, p, d)'$;
17: return $\beta$;
```

the valuations in the successor states. In the worst case, the number of valuations in the algorithm equals $N \cdot |A|^N$, as for the naive algorithm. However, if states contain identical successor states or $\gg$ can be used to reduce the number of valuations, then the effort is reduced. In the MDP-case, we only need to store one valuation per state, which implies that the effort equals the number of edges in the graph defined by $\mathcal{Y}_0, \ldots, \mathcal{Y}_N$.

A further improvement can be achieved if results are reused. Thus, we can define a lookup table in which $s \in \mathcal{Y}_n$ is stored together with the index $n$ and $s.\mathcal{V}$ during the backward phase. If, in a subsequent forward phase, state $s \in \mathcal{Y}_n$ is generated, $s.\mathcal{V}$ can be taken from the table, and no successors of $s$ need to be generated, since the evaluation for the backward phase is already available. That reuse of results should improve runtimes significantly for most models. If memory becomes a problem, only values from the upper levels $\mathcal{Y}_n$ ($n$ small) are stored, since their reuse introduces the largest benefits.

## *B. Transition Matrix Generation*

Since $\mathcal{X}$ contains the selected attack steps together with the transitions, it contains all information necessary for the analysis steps. Let $m$ be the cardinality of the state space. We describe the underlying stochastic process now in terms of vectors and matrices. The dimension of the vector is $1 \times m$ or $m \times 1$, and the order of the matrices is $m \times m$.

$\mathbf{P}$ is the transition matrix of the embedded discrete-time Markov chain (DTMC) that considers the state of the system when an attack step ends. We have

$$\mathbf{P}(s, s') = \sum_{o \in O_{\beta(s)}} \delta(E_{\beta(s)}(s, o) = s') Pr_{\beta(s)}(s, o),$$

where $\delta(b)$ for a Boolean expression is 1 for $b = true$ and

0 for $b = \mathit{false}$. Since $\mathbf{P}$ is usually not irreducible, the state space should be analyzed and reordered before the matrix is generated. The analysis is done using an algorithm, such as Tarjan's algorithm [11], to detect strongly connected components in the graph. Then $X$ can be decomposed into $X_0$, which is the set of transient states, and $X_1, \ldots, X_L$, which are $L$ irreducible subsets that are absorbing. If $X_0$ is nonempty, we generate $\mathbf{P}$ according to the decomposition and obtain the following structure.

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \cdots & \cdots & \mathbf{P}_{0,L} \\ \mathbf{0} & \mathbf{P}_{1,1} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \mathbf{0} & \mathbf{P}_{2,2} & \mathbf{0} & \vdots \\ \vdots & \ddots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{P}_{L,L} \end{pmatrix}$$

$\mathbf{P}_{0,0}$ is a substochastic matrix, and

$$\mathbf{N}_{0,0} = (\mathbf{I} - \mathbf{P}_{0,0})^{-1} = \sum_{k=0}^{\infty} (\mathbf{P}_{0,0})^k$$

exists and is nonnegative [2]. Moreover, $\mathbf{N}_{0,0}(s, s')$ equals the mean number of visits to state $s'$ before the set of transient states is left if the current state is $s$. $s_0$ belongs to $X_0$, and we can assume that it is the first state in the set. Often $\mathbf{P}_{0,0}$ has an additional structure such that $\mathbf{N}_{0,0}$ will not become a full matrix. However, this point will not be considered here. Using standard arguments from absorbing Markov chains [2], we find that the probability of entering subset $l$ from the initial state equals

$$pe_l = \mathbf{e}_1 \mathbf{N}_{0,0} \mathbf{P}_{0,l} \,\mathbb{1} = \mathbf{n}_{1*} \mathbf{P}_{0,l} \,\mathbb{1},$$

where $\mathbb{1}$ is a column vector of 1 of appropriate length. Since we have a unique initial state $s_0$, only the first row of matrix $\mathbf{N}$ is required, which results from the solution of $\mathbf{n}_{1*}(\mathbf{I} - \mathbf{P}_{0,0}) = \mathbf{e}_1$.

If $X_0$ is empty, then the set of states is irreducible, and we can use matrix $\mathbf{P}$ as it is; we assume that $s_0$ is the first state. Thus, the initial vector is defined as $\mathbf{e}_1$, which is a row vector in which the first element is 1 and all remaining elements are 0.

Matrix $\mathbf{P}$ does not distinguish between different attack steps. To introduce such a distinction, we define $\mathbf{P}^{\{i\}}$ as the matrix containing only transition probabilities according to $a_i \in A$.

$$\mathbf{P} = \sum_{a_i \in A} \mathbf{P}^{\{i\}}$$

Define $\mathbf{P}^{\neg\{i\}} = \mathbf{P} - \mathbf{P}^{\{i\}}$ and $\mathbf{p}^{\{i\}} = \mathbf{P}^{\{i\}} \,\mathbb{1}$. Of course, $\mathbf{P}^{\{i\}}$ and $\mathbf{P}^{\neg\{i\}}$ are structured like $\mathbf{P}$, but some of the nonzero submatrices in $\mathbf{P}$ may be zero in $\mathbf{P}^{\{i\}}$ or $\mathbf{P}^{\neg\{i\}}$. The notation can be extended to sets of attack steps or pairs of attack steps and outcomes. E.g., $\mathbf{P}^{\{(i,o),j\}}$ captures all transitions according to attack step $i$ with outcome $o$ and attack step $j$ with an arbitrary outcome.

## IV. Result Metrics and Their Analysis

Several example metrics have been defined in [3, Chapter 4.5]. We show how some metrics can be computed using analytical techniques that work on the state space. We also define several new metrics, which provide fresh insight.

The first class of metrics considers the adversary who achieves a goal or a set of goals. These measures can be defined for states. Thus, the state space can be decomposed into three subsets. $X_a$ is the set of states in which the adversary has already achieved his or her goal. $X_m$ is the subset in which the adversary may achieve his or her goal in the future. I.e., there exists a path from each state in $X_m$ to a state in $X_a$. Finally, $X_n$ is the set of states from which the adversary will never achieve his or her goal. I.e., from a state in $X_n$, no state in $X_a$ is reachable.

The three subsets can be computed with an extended algorithm for reachability analysis using the directed graph defined by matrix P. Initially $X_a$ contains all states where the adversary already achieved his or her goal. All states $s \in X$ that are not connected to any $s \in X_a$ are collected in $X_n$. Then all states $s \in X \setminus \{X_a \cup X_n\}$ without a connection to a state $s \in X_n$ are added to $X_a$, since from those states the adversary will eventually reach the goal. $X_m$ consists of the remaining states that are neither in $X_a$ nor in $X_n$.

If the initial state belongs to $X_a$ or $X_n$, analysis is not necessary. In the former case, the probability of reaching the required goals is 1, and the latter case, it is 0. To compute the probability if $s_0$ belongs to $X_m$, we make states in the sets $X_a$ and $X_n$ absorbing by removing all outgoing transitions and setting the probability of remaining in the state to 1. If we reorder the states according to the sets, $\mathbf{P}$ looks as follows.

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{m,m} & \mathbf{P}_{m,a} & \mathbf{P}_{m,n} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix}$$

The success probability corresponds to the absorption probability of the process in $X_a$, which equals

$$p_{succ} = \mathbf{e}_1 \sum_{k=0}^{\infty} (\mathbf{P}_{m,m})^k \mathbf{P}_{m,a} \,\mathbb{1} = \mathbf{e}_1 (\mathbf{I} - \mathbf{P}_{m,m})^{-1} \mathbf{P}_{m,a} \,\mathbb{1}.$$

Since all states in $X_m$ are connected to states in $X_a$, the above inverse matrix exists. Again, $\mathbf{e}_1(\mathbf{I} - \mathbf{P}_{m,m})^{-1}$ can be determined by solving a set of linear equations of order $|X_m|$.

Let $\mathbf{N}_{m,m} = (\mathbf{I} - \mathbf{P}_{m,m})^{-1}$. Then $\mathbf{N}_{m,m}(s, s')$ is the mean number of visits to state $s'$ before a state from $X_a \cup X_n$ is reached, under the condition that the current state is $s \in X_m$. To integrate timing information, we define a row vector $\mathbf{t}$ with $\mathbf{t}(s) = E[T_{\beta(s)}(s)]$. Consequently, $\mathbf{M}(s, s') = \mathbf{N}_{m,m}(s, s') \cdot \mathbf{t}(s')$ is the mean time spent in state $s'$ and $t^{abs} = \mathbf{e}_1 \mathbf{M} \,\mathbb{1}$ is the mean time to absorption. $t^{abs}$ is the mean time before the goal is achieved or it becomes clear that it will not be achieved.

To compute the time under the condition that the goal is eventually achieved, we have to consider the conditional absorbing Markov chain (see [6]). Let $\mathbf{p}_{m,a} = \mathbf{P}_{m,a} \,\mathbb{1}$ and define

$$\mathbf{u} = (\mathbf{I} - \mathbf{P}_{m,m})^{-1} \mathbf{p}_{m,a}.$$

$\mathbf{u}$ contains no 0 elements, since we have assumed that from every state in $X_m$ a state in $X_a$ is reachable. Then define $\mathbf{U} = diag(\mathbf{u})$, $\mathbf{P}^C_{m,m} = \mathbf{U}^{-1}\mathbf{P}_{m,m}\mathbf{U}$, and $\mathbf{p}^C_{m,a} = \mathbf{U}^{-1}\mathbf{p}^C_{m,a}$. Matrix $\mathbf{P}^C_{m,a}$ and vector $\mathbf{p}^C_{m,a}$ describe a new absorbing Markov chain with

$$\mathbf{N}^C_{m,m} = (\mathbf{I} - \mathbf{P}^C_{m,m})^{-1} = \mathbf{U}^{-1}\mathbf{N}_{m,m}\mathbf{U}$$

such that $\mathbf{N}^C_{m,m}(s, s')$ contains the mean number of visits to state $s'$ under the condition that the current state is $s$ and absorption occurs in a state from $X_a$. Consequently, we can use $\mathbf{N}^C_{m,m}$ rather than $\mathbf{N}_{m,m}$ to compute the mean time to a successful attack by defining $\mathbf{M}^C(s, s') = \mathbf{N}^C_{m,m}(s, s') \cdot \mathbf{t}(s')$ and $t^{abs}_a = \mathbf{e}_1\mathbf{M}^C\,\mathbb{1}$ as the mean time of a successful attack under the condition that the attack is successful.

The computation of cost- and gain-related measures is similar to the computation of mean times. Thus, define

$$\mathbf{C}(s, s') = \mathbf{N}_{m,m}(s, s')C_{\beta(s)}(s).$$

Then $c^{abs} = \mathbf{e}_1\mathbf{C}\,\mathbb{1}$ is the average cost before absorption, i.e., before the goal states have been reached or become unreachable. The costs of a successful attack are $c^{abs}_a = \mathbf{e}_1\mathbf{C}^C\,\mathbb{1}$, where $\mathbf{C}^C$ is built like $\mathbf{C}$ using $\mathbf{N}^C_{m,m}$ rather than $\mathbf{N}_{m,m}$.

Computation of the gain is again similar; the only difference arises from the gain's association with transitions rather than states. Thus, define

$$\mathbf{g}_m(s) = \sum_{s' \in X_m} \left( \sum_{(i,o):a_i=\beta(s)\wedge s \xrightarrow{i,o} s'} Pr_i(s, o)G_i(s, o, s') \right).$$

Then

$$\sum_{s' \in X_m} \mathbf{N}_{m,m}(s_0, s')\mathbf{g}_m(s')$$

is the average gain due to attack steps that do not leave $X_m$. To compute the average gain until a state outside $X_m$ is reached, the gain of the transition that leaves $X_m$ must also be considered. The probability of leaving $X_m$ from state $s \in X_m$ such that $s' \in X_a \cup X_n$ equals $\mathbf{N}_{m,m}(s_0, s)\mathbf{P}(s, s')$. That implies that we obtain $g^{abs}$ for the average gain until a state in $X_n \cup X_a$ is reached.

$$g'_m(s') = \sum_{s'' \in X_a \cup X_n} \left( \sum_{(i,o):a_i=\beta(s')\wedge s' \xrightarrow{i,o} s''} Pr_i(s', o)G_i(s', o, s'') \right)$$

$$g^{abs} = \sum_{s' \in X_m} \mathbf{N}_{m,m}(s_0, s')\left(\mathbf{g}_m(s') + g'_m(s')\right)$$

Again, the conditional gain if a state from $X_a$ is entered can be computed using $\mathbf{N}^C_{m,m}$ in the previous equations.

For the computation of the probability that an attack step is not detected, let $\mathbf{F}$ be an $m \times m$ matrix defined as follows:

$$\mathbf{F}(s, s') = \sum_{(i,o):a_i=\beta(s)\wedge s \xrightarrow{i,o} s'} Pr_i(s, o)F_i(s, o).$$

Matrix $\mathbf{F}$ includes the probabilities that an attack step is performed and not detected. $\mathbf{F}$ can be decomposed like $\mathbf{P}$, and since $\mathbf{F}(s, s') \leq \mathbf{P}(s, s')$, the inverse matrix $(\mathbf{I} - \mathbf{F}_{m,m})^{-1}$ exists and

$$p_{ndet} = \mathbf{e}_1(\mathbf{I} - \mathbf{F}_{m,m})^{-1}\mathbf{P}_{m,a}\,\mathbb{1}$$

is the probability that the adversary reaches a state in $X_a$ without being detected.

Another class of metrics is related to the occurrence of specific attack steps or outcomes of attack steps in a sequence of attacks. To determine these measures, we use the matrices $\mathbf{P}^{\{i\}}$ and $\mathbf{P}^{\neg\{i\}}$. In order to avoid notation overload, from here on, we use $\mathbf{P}^{\{i\}}$ for probabilities related to specific attack steps and also to attack step outcome pairs for which the index $(i, o)$ would be more appropriate or to sets of attack steps and outcomes. If the state is irreducible and $\mathbf{P}^{\{i\}} \neq \mathbf{0}$, then the probability of performing attack step $i$ is 1. Otherwise, the probability of performing attack step $i$ in a transient state equals $pa^i_0 = 1 - \mathbf{e}_1(\mathbf{I} - \mathbf{P}^{\neg\{i\}}_{0,0})^{-1}\sum_{l=1}^{L}\mathbf{P}^{\neg\{i\}}_{0,l}\,\mathbb{1}$, and the probability of performing attack step $i$ in $X_l$ $(l = 1, \ldots, L)$ under the condition that it has not been performed in a state from $X_0$ equals 0 if $\mathbf{P}^{\{i\}}_{l,l} = \mathbf{0}$ and equals $pa^i_0 = \mathbf{e}_1(\mathbf{I} - \mathbf{P}^{\neg\{i\}}_{0,0})^{-1}\mathbf{P}^{\neg\{i\}}_{0,l}\,\mathbb{1}$ otherwise, such that the probability of performing attack step $i$ is given by $pa^i = \sum_{l=0}^{L} pa^i_l$. The computations can be extended to sequences of attack steps and attack step outcome pairs. E.g., the probability of performing attack step $i$ before $j$ is given by

$$\mathbf{e}_1\sum_{k=0}^{\infty}(\mathbf{P}^{\neg\{i,j\}})^k\mathbf{P}^{\{i\}}\sum_{k=0}^{\infty}(\mathbf{P}^{\neg\{j\}})^k\mathbf{P}^{\{j\}}\,\mathbb{1}.$$

If all states in matrices $\mathbf{P}^{\neg\{i,j\}}$ or $\mathbf{P}^{\neg\{j\}}$ are transient, then the infinite matrix sums can be replaced by the inverse matrices $(\mathbf{I}-\mathbf{P}^{\neg\{i,j\}})^{-1}$ and $(\mathbf{I}-\mathbf{P}^{\neg\{j\}})^{-1}$ if appropriate.

We can, of course, combine the probability measures related to attacks or attack sequences and the cost/time/gain results that have been described with respect to entering a set of states. Concrete approaches can be developed according to specific measures.

For stationary analysis, one has to compute the stationary distribution inside each irreducible subset; it is then multiplied by the probability of reaching the subset from the initial state, i.e., by probability $pe_l$. The embedded stationary distribution of the states in subset $l$ is given by the solution of

$$\mathbf{p}_l = \mathbf{p}_l\mathbf{P}_{l,l} \text{ and } \mathbf{p}_l\,\mathbb{1} = pe_l.$$

The embedded stationary distribution does not consider the different sojourn times due to different values for $T(s)$. The stationary vector can be achieved by normalizing the values of the embedded stationary distribution according to the sojourn times for the states in the irreducible subset. The components of the stationary vector $\pi$ are computed as

$$\pi_l(s) = \frac{\mathbf{p}_l(s)\mathbf{t}(s)}{\sum_{s' \in X_l}\mathbf{p}_l(s')\mathbf{t}(s')}.$$

The frequency of attack step $i$ is then given by

$$f_i = \sum_{l=1}^{L} \sum_{s \in X_l} \frac{\pi_l(s) \sum_{s' \in X_l} \mathbf{P}^{\{i\}}(s, s')}{T(s)}.$$

The metrics computed up to now are all related to an infinite time horizon on which the time might be stopped when a specific condition is observed, i.e., a state has been reached or an attack has occurred. The situation is more complex if the system should be analyzed for a finite time horizon $\tau$. In that case, transient analysis has to be performed. We briefly consider five different approaches, all of which have specific advantages and disadvantages.

First, if $\mathbf{P}$ can be reordered to an upper triangular matrix, then the system is completely acyclic. In that case, the behavior of the system is described by a finite set of paths, and a *path-based analysis* can be performed. Each path starts in the initial state $s_0$ and ends after finitely many steps in an absorbing state. For a path, the path probability and the probability of being in a specific state at time $\tau$ can be computed. However, computation of the latter requires the computation of convolutions of the distributions $T_i$ along the path, which is cumbersome for longer paths.

The second approach is to substitute transition timing distributions by geometric distributions with the same mean. Define $0 < \Delta \leq \min_{s \in X}(T_{\beta(s)}(s)/(1 - \mathbf{P}(s,s)))$ as the basic time step of the new discrete-time Markov chain. Then define a new transition matrix $\overline{\mathbf{P}}$ as

$$(1 - \overline{\mathbf{P}}(s,s)) = (1 - \mathbf{P}(s,s)) \frac{\Delta}{T_{\beta(s)}(s)}$$

which implies $0 \leq \overline{\mathbf{P}}(s,s) < 1$. Then define

$$\overline{\mathbf{P}}(s,s') = \mathbf{P}(s,s') \frac{\Delta}{T_{\beta(s)}(s)} \text{ for } s \neq s'.$$

which implies

$$\sum_{s' \in X} \overline{\mathbf{P}}(s,s') = \sum_{s' \in X \setminus \{s\}} \overline{\mathbf{P}}(s,s') + \overline{\mathbf{P}}(s,s) = \\ 1 - (1 - \mathbf{P}(s,s)) \tfrac{\Delta}{T_{\beta(s)}(s)} + (1 - \mathbf{P}(s,s)) \tfrac{\Delta}{T_{\beta(s)}(s)} = 1.$$

*DTMC analysis* can be performed using matrix $\overline{\mathbf{P}}$, in which every step has a duration of $\Delta$ such that the distribution of the DTMC at discrete time points $k \cdot \Delta$ can be computed.

Similarly, one can use continuous-time Markov chains (CTMCs) for analysis. In that case, each sojourn time is replaced by an exponential distribution with rate $\mu_s = (E[T_{\beta(s)}(s)])^{-1}$. The resulting model is a CTMC on the same state space $X$, and standard *CTMC analysis* can be applied.

The approach may be extended through the representation of each sojourn time $T_{\beta(s)}(s)$ as a PH distribution. E.g., fitting the first two moments results in a PH distribution with 2 phases, if the coefficient of variation is not too small. In that case, the state space of the CTMC is enlarged, since the phase of the distribution has to be considered in addition to the state from $X$. If each sojourn time is replaced with a PH distribution with 2 phases, the size of the state space doubles. *Extended CTMC analysis* can then use standard techniques for the transient analysis of CTMCs.

The final analysis method is *simulation*, which can easily be performed using matrix $\mathbf{P}$ and $T_{\beta(s)}(s)$ for all $s \in X$. Since the values for $\beta(s)$ are available, they do not need to be recomputed during simulation, which is therefore more efficient.

## V. Scalability and Timing

While the types of numerical analysis we demonstrate can significantly reduce the analysis runtime when compared to the simulation approach used in [3] and [4], they require full exploration of the reachable state space. In this section, we present timing measurements to assess the scalability of the state-space generation algorithm and the attack step selection algorithm.

### A. Scalability

First, we present an example model that scales easily in both the state space and the reachable state space. The attack execution graph for this "gatekeeper model" is shown in Figure 2. The AEG consists of a single initial state variable that enables the entire row of attack steps. The successful outcome for each attack step captures the corresponding state variable in the row below. One final attack step is enabled by possessing any of the state variables in the state variable row, and a successful outcome results in achieving the goal. Observe that although the adversary can be successful in only two steps, he or she may need many more steps, since attack steps may not be successful.

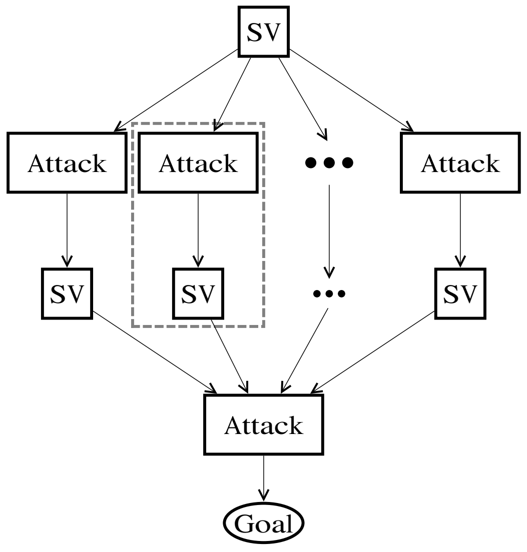


Figure 2. Gatekeeper Model

We took timing measurements while varying two factors in the model. One was the number of attack step and state variable node pairs in the center rows of the AEG. (One node pair is circled in the figure for clarity.) The other factor was the look-ahead of the adversary in the adversary profile. The size of the state-space is $2^{(N+2)}$, where $N$ is the number of nodes in the AEG. More importantly, the complexity of the determine_beta function is on the order of $O((2N)^L)$, where $L$ represents the look-ahead. For each configuration, we measured the overall time spent to generate the state space. The results are shown in Table II.

Table II
STATE-SPACE GENERATION TIMING (IN SECONDS)

| Look-ahead | 5 Nodes | 10 Nodes | 20 Nodes | 40 Nodes |
|---|---|---|---|---|
| 3 | 0.002 | 0.036 | 0.513 | 15.16 |
| 4 | 0.022 | 0.577 | 16.128 | 1003 |
| 5 | 0.174 | 8.885 | 569.98 | - |
| 6 | 1.336 | 149.20 | - | - |
| 7 | 10.131 | - | - | - |

The time to generate the state space followed the general exponential formula expressed earlier in the paper. The time increased significantly as the look-ahead and the number of available attack steps increased. Some configurations took too long to compute within reasonable bounds.

However, those results were generated from a model in which the reachable state space was almost as large as the overall state space. In typical models, that would not be the case. There is an inherent ordering to attack steps that prevents the reachable state space from being anywhere near the size of the full state space.

### B. Performance of Real-World Models

We repeated the timing experiments on the two SCADA models used in [4] in order to demonstrate that the numerical methods produce the same results as simulation, that the numerical methods are faster than simulation, and that the numerical runtime is fast for typical models.

Each of the two SCADA models represents a corporate network and a control network that communicate through a firewall. In the non-DMZ model, a data historian is located within the corporate network, and direct communication is allowed through the firewall. In the DMZ model, the data historian is placed in a demilitarized zone between the two networks. There, the firewall blocks direct communication, instead forcing any communication between the networks to pass through the data historian.

The non-DMZ model consists of 18 attack steps, eight access domains, four knowledge items, and five attack goals, for a total of 19 state variables. The DMZ model has an additional two attack steps and an additional access domain for a total of 20 state variables. Note that the adversaries we define for analysis typically have a look-ahead of 3 or 4 steps.

Table III contains timing measurements for both simulation and state-space generation of the two typical models. The numerical solution techniques significantly outperform the simulations in all cases, mostly because of the repetitive nature of simulation.

In comparison to the state-space generation times of our Gatekeeper model, real-world models with the same number of state variables take considerably less time to complete. First, we note that the size of the reachable state space is very small compared to the size of the full state space for real models. This in turn reduces the number of determine_beta computations that must be performed. Also, although the DMZ model has one more state variable, its reachable state space is smaller than that of the non-DMZ model, showing that the structure of the model has a significant impact on the timing. Finally, and most importantly, the state-space generation completes quickly for typical models.

Table III
SIMULATION AND NUMERICAL SOLUTION TIMING (IN SECONDS)

| | Simulation | | Numerical | |
|---|---|---|---|---|
| Adversary | Non-DMZ | DMZ | Non-DMZ | DMZ |
| Nation-State | 121.94 | 74.99 | 0.009 | 0.004 |
| Lone Hacker | 123.17 | 5.94 | 0.008 | 0.001 |
| Terrorist Org | 40.63 | 18.90 | 0.017 | 0.005 |
| Employee | 6.58 | 5.09 | 0.002 | 0.001 |
| Sys Admin | 75.10 | 55.65 | 0.021 | 0.010 |

## VI. EXAMPLE METRICS

In this section, we present example metrics from the DMZ and non-DMZ models. We compare these metrics to the metrics presented in [4] in order to validate our method, and we present new metrics to provide further insight.

First, we consider the time that the adversary takes to reach a goal. This is the metric that was reported in [4] using simulation. Table IV contains the resulting metric mean and 95% confidence interval range from a simulation over 500 runs as well as the result of the numerical analysis. Note that the simulation results are imprecise due to the confidence interval range when compared to the results from numerical analysis. Additionally, as shown in Table IV, two values did not converge within the specified number of simulation runs. The precision and the short computation time are significant advantages of the numerical analysis.

Table IV
TIME TO GOAL (IN MINUTES)

| | Simulation | | Numerical | |
|---|---|---|---|---|
| Adversary | Non-DMZ | DMZ | Non-DMZ | DMZ |
| Nation-State | $116.3 \pm 3.58$ | $209.8 \pm 5.2$ | 114.3 | 209.3 |
| Lone Hacker | $116.3 \pm 3.58$ | - | 114.3 | - |
| Terrorist Org | $348.9 \pm 13.8$* | $353.1 \pm 20.1$* | 370. | 370. |
| Employee | $15.04 \pm 0.48$ | - | 15.3 | - |
| Sys Admin | $15.72 \pm 0.76$ | $15.2 \pm 1.5$ | 15.29 | 15.29 |

* Did not converge.

Next, we present two new metrics: the frequency of specific attacks within a sequence of attacks, and the probability of detecting an attack. Computing the frequency of an attack step is explored in [3] using simulation. However, in order to enable calculation of the frequency of an attack step during simulation, a reward variable must be created for each individual attack step. Then, a reward event is fired each time an attack step is selected. Based on the total number of attack step reward events, the attack step frequency is manually computed.

When numerical methods are used, once the probability transition matrix has been generated, the attack step frequency metric can quickly be computed for all attack steps in a single operation. Recall that timings for simulation and numerical solutions are presented in Table III.

Table V
FREQUENCY OF ATTACK STEPS

| Adversary | Non-DMZ | | DMZ | |
|---|---|---|---|---|
| Nation-State | Hack User Acct. | 87.5% | Hack User Acct. | 57.3% |
| | Access Data | 12.5% | Hack Data Server | 35.8% |
| | | | Access Data | 6.8% |
| Lone Hacker | Hack User Acct. | 87.5% | Do Nothing | 100% |
| | Access Data | 12.5% | | |
| Terrorist Org | Hack User Acct. | 32.4% | Hack User Acct. | 32.4% |
| | Elevate Privilege | 13.5% | Elevate Privilege | 13.5% |
| | Hack Ctrl. Server | 54.0% | Hack Ctrl. Server | 54.0% |
| Employee | User Login | 6.5% | Do Nothing | 100% |
| | Access Data | 93.5% | | |
| Sys Admin | Admin Login | 6.5% | Admin Login | 6.5% |
| | Access Data | 93.5% | Access Data | 93.5% |

Table V reports the percentage of time spent attempting specific attack steps. Here, it is once again notable that adding the DMZ to the system deters both the Lone Hacker and the Employee adversaries from attempting any attack whatsoever. Also, the Nation-State adversary clearly changes its route to the goal. In addition to tracing the paths of adversaries through the system, the frequency of attack steps is also useful for determining the effects of deterrent mechanisms on the time an adversary spends on a given attack.

Table VI
PROBABILITY OF DETECTION

| Adversary | Non-DMZ | DMZ |
|---|---|---|
| Nation-State | 0.0784516 | 0.220143 |
| Lone Hacker | 0.0784516 | - |
| Terrorist Org | 0.478188 | 0.478188 |
| Employee | 0.0241125 | - |
| Sys Admin | 0.0241125 | 0.0635423 |

Finally, we present the probability of detection in Table VI. This metric is simple to compute using numerical analysis; however, it is not supported by the simulation tool we used. We note that the Terrorist Organization attacker is much more likely to be detected than any other attacker. Furthermore, by adding the demilitarized zone to the network, we triple the probability of detecting the Nation-State and System Administrator adversaries.

## VII. CONCLUSION

In this paper, we extended the capabilities of the ADVISE method to quantitatively analyze attacks on large computer systems. In the previous version of ADVISE, analysis was made by simulation, and the decisions of adversaries were restricted to policies that depend only on the current state. In order to remove limitations on the adversary utility functions, we have formally defined the attack step selection with respect to a history-consistent policy. We can then use a partial ordering to reduce the problem size. Finally, we introduced a general algorithm for exploring the reachable state space and generating the transition matrix. Once the transition matrix is available, numerous metrics can be generated through numerical analysis. Furthermore, the transition matrix can be used to facilitate other forms of analysis. We have demonstrated the practical use of our work by timing the process and have provided insight through a case study using new metrics.

## ACKNOWLEDGMENT

The work described in this paper was conducted, in part, with funding from the Department of Homeland Security under contract FA8750-09-C-0039 with the Air Force Research Laboratory. We particularly wish to thank Douglas Maughan, Director of the Cyber Security Division in the Homeland Security Advanced Research Projects Agency (HSARPA), within the Science and Technology (S&T) Directorate of the Department of Homeland Security (DHS). The authors would also like to thank Jenny Applequist for her editorial work. Finally, we wish to thank the reviewers for their insightful comments.

## REFERENCES

[1] S. Evans and J. Wallner. Risk-based security engineering through the eyes of the adversary. In *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, pages 158–165. IEEE, 2005.

[2] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Springer, 1976.

[3] E. A. LeMay. *Adversary-Driven State-Based System Security Evaluation*. PhD thesis, University of Illinois at Urbana-Champaign, 2011.

[4] E. A. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrke. Model-based security metrics using ADversary VIew Security Evaluation (ADVISE). In *Quantitative Evaluation of Systems (QEST)*, pages 191–200. IEEE, 2011.

[5] R. P. Lippmann. An annotated review of past papers on attack graphs. Technical report, DTIC Document, 2005.

[6] P. Narain. The conditional Markov chain in a genetic context. *Journal of Genetics*, 63(2):49–62, 1977.

[7] D. M. Nicol, W. H. Sanders, and K. S. Trivedi. Model-based evaluation: From dependability to security. *IEEE Trans. on Dependable and Secure Computing*, 1(1):48–65, 2004.

[8] R. Ortalo, Y. Deswarte, and M. Kaâniche. Experimenting with quantitative evaluation tools for monitoring operational security. *IEEE Trans. on Software Engineering*, 25(5):633–650, 1999.

[9] M. L. Puterman. *Markov Decision Processes*. Wiley, 2005.

[10] A. Singhal and X. Ou. Security risk analysis of enterprise networks using probabilistic attack graphs. Technical Report NISTIR 7788, National Institute of Standards and Technology, September 2011.

[11] R. E. Tarjan. Depth-first search and linear graph-algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[12] B. Whiteman. Network Risk Assessment Tool (NRAT). *IA Newsletter*, 11(1):4–8, Spring 2008.