Multi-Stage Attack Graph Security Games: Heuristic Strategies, with Empirical Game-Theoretic Analysis

Thanh H. Nguyen, Mason Wright, Michael P. Wellman, Satinder Singh
University of Michigan, Ann Arbor
{thanhhng,masondw,wellman,baveja}@umich.edu

ABSTRACT

We study the problem of allocating limited security countermeasures to protect network data from cyber-attacks, for scenarios modeled by Bayesian attack graphs. We consider multi-stage interactions between a network administrator and cybercriminals, formulated as a security game. This formulation is capable of representing security environments with significant dynamics and uncertainty, and very large strategy spaces. For the game model, we propose parameterized heuristic strategies for both players. Our heuristics exploit the topological structure of the attack graphs and employ different sampling methodologies to overcome the computational complexity in determining players' actions. Given the complexity of the game, we employ a simulation-based methodology, and perform empirical game analysis over an enumerated set of these heuristic strategies. Finally, we conduct experiments based on a variety of game settings to demonstrate the advantages of our heuristics in obtaining effective defense strategies which are robust to the uncertainty of the security environment.

KEYWORDS

Game Theory; Bayesian Attack Graph; Moving Target Defense

1 INTRODUCTION

Cyber-attacks on computer networks are a critical threat which has been rapidly increasing and becoming more sophisticated. According to the U.S. Government Accountability Office's survey of 24 federal agencies, the number of cyber-attack incidents rose thirteenfold from about 5,500 in 2006 to over 75,000 in 2015 [19]. These attacks can lead to serious economic and security consequences such as sensitive data loss and disruption of services provided by critical infrastructure. Various solutions for enhancing network security have been provided such as deploying firewalls, finding and patching vulnerabilities, and detecting and preventing intrusions [2, 21, 23].

Attack graphs are a graphical model form, employed as a tool in cybersecurity research to hierarchically decompose complex security scenarios into simple and quantifiable actions [3, 4, 12, 17]. Many research efforts have used attack graphs or related forms of graphical security models to analyze different complex security scenarios [12]. In particular, attack graph models have been used as a tool for evaluating network hardening strategies, in which a network administrator (the defender) deploys security countermeasures to protect network data from cybercriminals (the attacker) [1, 5–7, 11, 15, 16, 25].

Attack graphs are particularly suitable for modeling scenarios in moving target defense (MTD) [10], where the defender employs proactive tactics to dynamically change network configurations to limit the exposure of vulnerabilities. The advantages of MTD techniques are most salient for thwarting progressive attacks [8, 24], as reconfiguration prevents the attacker from exploiting knowledge accumulated over time. Attack graphs naturally represent progress of an attack, and the defense actions in our model, here defined abstractly, may incorporate MTD methods or other measures.

We build on prior work representing a variety of security problems with attack graphs, and take a game-theoretic approach to reason about the strategic interaction between the defender and the attacker. Building on an existing Bayesian attack graph formalism [15, 18], we model the problem as a simultaneous multi-stage attack-graph security game. Nodes in a Bayesian attack graph represent security conditions of the network system. For example, an SSH buffer overflow vulnerability in an FTP server can be considered a security condition, as could user privileges achieved as a result of exploiting that vulnerability. The defender attempts to protect a set of goal nodes (critical security conditions) in the attack graph. Conversely, the attacker, starting from some initial security conditions, follows paths through the graph to undermine these goal nodes. At every time step, both the defender and the attacker simultaneously take actions. Given limited security resources, the defender has to decide for which nodes of the attack graph to deploy security countermeasures. Meanwhile, the attacker selects nodes to attack in order to progress toward the goals. The outcome of the players' actions (whether the attacker succeeds) follows a stochastic process, which represents the success probability of the actions taken. These outcomes are only imperfectly observed by the defender, adding further uncertainty which must be taken account in its strategic reasoning.

Based on our game model, we propose different parameterized strategies for both players. At each time step, our attack strategies assess the value of each possible attack action, by examining possible attack paths. These paths are sequences of nodes which could feasibly be attacked in future time steps (as a result of attack actions in the current time step) in order to reach goal nodes. Since there are exponentially many possible attack paths, it is impractical to evaluate all of them. Therefore, we introduce two heuristics. First, we estimate the attack value for each individual node locally, based on the attack values of neighboring nodes. Attack values of goal nodes, in particular, correspond to the importance of each goal node.

MTD'17, October 30, 2017, Dallas, TX, USA.
 ISBN 978-1-4503-5176-8/17/10…$15.00
DOI: https://doi.org/10.1145/3140549.3140562

Second, we consider only a small subset of attack paths, selected by random sampling, according to the likelihood the attacker will successfully reach a goal node by following each path.

At each time step, our defense strategies have two stages: (i) update the defender's belief about the outcome of players' actions in the previous time step, and (ii) generate a new defense action based on the updated belief and the defender's assumption about the attacker's strategy. For stage (i), we apply particle filtering [20] to deal with the exponential number of possible outcomes. For stage (ii), we evaluate defense candidate actions using concepts similar to those employed in the attack strategies.

Finally, we employ a simulation-based methodology, called *empirical game-theoretic analysis* (EGTA) [22], to construct and analyze game models over the heuristic strategies. We present detailed evaluation of the proposed strategies based on this game analysis. Our experiments are conducted over a variety of game settings with different attack graph topologies. We show that our defense strategies provide high solution quality compared to multiple baselines. Furthermore, we examine the robustness of defense strategies to uncertainty about game actions and to the attacker's strategy selection.

2 RELATED WORK

Attack graphs are commonly used to provide a convenient representation for analysis of network vulnerabilities [3, 4, 17]. In an attack graph, nodes can represent conditions on attack states of a system, and edges can represent relationships among these conditions: specifically, how the achievement of specific conditions through an attacker's actions can enable other conditions. *Bayesian attack graphs*, proposed later by Liu and Man, combine attack graphs with quantified uncertainty on attack states and relations [13]. Revised versions of Bayesian attack graphs were then introduced which incorporate other security aspects such as dynamic behavior and mitigation strategies [9, 18]. In fact, several different graph-based security models with some analogy to attack graphs have been proposed to represent and analyze complex security scenarios [12]. Our work is based on a specific formulation of Bayesian attack graphs, detailed below. The basic ideas of our game model and heuristic strategies would also apply to variant forms of graph-based security models with reasonable modifications.

Given the prominence of graph-based security models, previous work has proposed different game-theoretic solutions for finding an optimal defense policy based on those models. Durkota et al. study the problem of hardening security of a network system by deploying honeypots to the network to deceive the attacker [6, 7]. They model the problem as a Stackelberg security game in which the attacker's plans are compactly represented using attack graphs. Zonouz et al. present an automated intrusion response system which models the security problem on an attack-response tree as a two-player zero-sum Stackelberg stochastic game [25]. Besides Stackelberg games, single-stage simultaneous games are also applied to model the security problem on attack-defense trees, and Nash equilibrium is used to find an optimal defense policy [1, 5, 11].

Our specific game-theoretic model is built on the attack graph formalism of Miehling et al. [15]. In their work, the attacker's behavior is modeled by a probabilistic spreading process, which is known by the defender. In our model, both the defender and the attacker dynamically decide on which actions to take at every time step, depending on their knowledge with respect to the game.

3 GAME MODEL

3.1 Game definition

Definition 3.1 (Bayesian Attack Graph [9, 15]). A *Bayesian attack graph* is a directed acyclic graph, denoted by $\mathbf{G} = (\mathbf{V}, s_0, \mathbf{E}, \theta, p)$.

- $\mathbf{V}$ is a non-empty set of nodes, representing security conditions of a system.
- At time t, each node v has a *state* $s_t(v) \in \{0, 1\}$, where 0 means v is *inactive* (i.e., a security state not compromised by the attacker) and 1 means it is *active* (compromised). The *initial state* $s_0(v)$ represents the initial setting of node v.
- $\mathbf{E}$ is a set of directed edges between nodes in $\mathbf{V}$, each edge representing an *atomic attack action* or an *exploit*. For edge $e = (u, v) \in \mathbf{E}$, u is called a *precondition* and v is called a *postcondition*. We denote by $\pi^-(v) = \{u \mid (u, v) \in \mathbf{E}\}$ the set of preconditions and $\pi^+(v) = \{u \mid (v, u) \in \mathbf{E}\}$ the set of postconditions associated with node $v \in \mathbf{V}$. An exploit $e = (u, v) \in \mathbf{E}$ is *feasible* when its precondition u is active.
- Nodes $\mathbf{V}^r = \{v \in \mathbf{V} \mid \pi^-(v) = \emptyset\}$ without preconditions are called *root nodes*. Nodes $\mathbf{V}^l = \{v \in \mathbf{V} \mid \pi^+(v) = \emptyset\}$ without postconditions are called *leaf nodes*.
- Each node $v \in \mathbf{V}$ is assigned a *node type* $\theta(v) \in \{\vee, \wedge\}$. An $\vee$-type node v can be activated via any of the feasible exploits into v. Activating an $\wedge$-type node v requires all exploits into v to be feasible and taken. Root nodes ($\wedge$-type nodes without preconditions) have no prerequisite exploits, and so can be activated directly. We denote by $\mathbf{V}^\wedge$ the set of all $\wedge$-type nodes and $\mathbf{V}^\vee$ the set of all $\vee$-type nodes. The sets of edges into $\vee$-type nodes and $\wedge$-type nodes, respectively, are denoted by $\mathbf{E}^\vee = \{(u, v) \in \mathbf{E} \mid v \in \mathbf{V}^\vee\}$ and $\mathbf{E}^\wedge = \{(u, v) \in \mathbf{E} \mid v \in \mathbf{V}^\wedge\}$.
- The *activation probability* $p(e) \in (0, 1]$ of edge $e = (u, v) \in \mathbf{E}^\vee$ represents the probability the $\vee$-node v becomes active when the exploit e is taken. $\wedge$-type nodes are also associated with activation probabilities; $p(v) \in (0, 1]$ is the probability $v \in \mathbf{V}^\wedge$ becomes active when all exploits into v are taken.

Definition 3.2 (Attack Graph Game). An attack graph security game is defined on a Bayesian attack graph $\mathbf{G} = (\mathbf{V}, s_0, \mathbf{E}, \theta, p)$ by elements $\mathbf{\Psi} = (\mathbf{T}, \mathbf{V^g}, \mathbf{S}, \mathbf{O}, \mathbf{D}, \mathbf{A}, \mathbf{R}, \mathbf{C})$:

- **Time step**: $\mathbf{T} = \{0, \ldots, T\}$ where T is the time horizon.
- **Player goal**: A non-empty subset $\mathbf{V}^g \subseteq \mathbf{V}$ of nodes are distinguished as *critical* security conditions. The attacker aims to activate these *goal* nodes while the defender attempts to keep them inactive.
- **Graph state**: $\mathbf{S} = \{\mathbf{S}_0, \ldots, \mathbf{S}_T\}$ where $\mathbf{S}_t = \{v \in \mathbf{V} \mid s_t(v) = 1\}$ represents the active nodes at time step t.
- **Defender observation**: $\mathbf{O} = \{\mathbf{O}_0, \ldots, \mathbf{O}_T\}$, where $\mathbf{O}_t$ associates each node v with one of the signals $\{0_v, 1_v\}$. Signal 1_v (0_v) indicates v is active (inactive). If v is active at t, signal 1_v is generated with probability $p(1_v \mid s_t(v) =$

Algorithm 1: State transition for node v, according to $T(\mathbf{S}_t, \mathbf{A}_{t+1}, \mathbf{D}_{t+1})$.

```
Initialize s_{t+1}(v) ← s_t(v);
if v ∈ D_{t+1} then
    s_{t+1}(v) ← 0;          // defender overrules attacker
else
    if v ∈ A_{t+1} ∩ V^∧ then
        with probability p(v), s_{t+1}(v) ← 1
    else
        for (u, v) ∈ A_{t+1} ∩ E^∨ and s_{t+1}(v) = 0 do
            with probability p(u, v), s_{t+1}(v) ← 1
```

$1) \in (0, 1]$, and if v is inactive, signal 1_v is generated with probability $p(1_v \mid s_t(v) = 0) < p(1_v \mid s_t(v) = 1)$. Otherwise, signal 0_v is generated. The signals are independently distributed, over time and nodes.

- **Player action**: $\mathbf{D} = \{\mathbf{D}_0, \ldots, \mathbf{D}_T\}$ where defender action $\mathbf{D}_t \subseteq \mathbf{V}$ comprises a set of nodes which the defender *disables* at time step t. $\mathbf{A} = \{\mathbf{A}_0, \ldots, \mathbf{A}_T\}$ where attacker action $\mathbf{A}_t \subseteq \mathbf{V}^\wedge \cup \mathbf{E}^\vee$ consists of (i) $\wedge$-type nodes $v \in \mathbf{V}^\wedge$, meaning that the attacker takes all exploits into v to activate that node at time step t and (ii) exploits into $\vee$-type nodes $(u, v) \in \mathbf{E}^\vee$, meaning that attacker takes exploit (u, v) to activate the $\vee$-type node v at time step t.
- **Goal reward**: $\mathbf{R}$ assigns a reward for the players to each goal node $v \in \mathbf{V}^g$. $r^a(v) > 0$ is the attacker reward and $r^d(v) < 0$ is the defender reward (i.e., a penalty) if v is active. For inactive goal nodes, both receive zero.
- **Action cost**: $\mathbf{C}$ assigns a cost to each action the players take. In particular, $c^a(e) < 0$ is the attacker's cost to attempt exploit $e \in \mathbf{E}^\vee$ and $c^a(v)$ is the attacker's cost to attempt all exploits into $\wedge$-type node $v \in \mathbf{V}^\wedge$. The defender incurs cost $c^d(v) < 0$ to disable node $v \in \mathbf{V}$.
- **Discount factor**: $\gamma \in (0, 1]$.

Initially, $\mathbf{D}_0 \equiv \emptyset$, $\mathbf{A}_0 \equiv \emptyset$, and $\mathbf{S}_0 \equiv \emptyset$. We assume the defender knows only the initial graph state $\mathbf{S}_0$, whereas the attacker is fully aware of graph states at every time step. Thus, we can set $\mathbf{O}_0 = \emptyset$. At each time step $t+1 \in \{1, \ldots, T\}$, the attacker decides which feasible exploits to attempt. At time step $t+1 = 1$, in particular, the attacker can choose any root nodes $v \in \mathbf{V}^r$ to activate directly with a success probability $p(v)$. Simultaneously, the defender decides which nodes to disable to prevent the attacker from intruding further.

3.2 Timing of game events

The game proceeds in discrete time steps, $t + 1 \in \{1, \ldots, T\}$, with both players aware of the current time. At each time step $t + 1$, the following sequence of events occurs.

(1) Observations:
   - The attacker observes $\mathbf{S}_t$.
   - The defender observes $\mathbf{O}_t \sim \mathbf{O}(\mathbf{S}_t)$.

(2) The attacker and defender simultaneously select actions $\mathbf{A}_{t+1}$ and $\mathbf{D}_{t+1}$ according to their respective strategies.

(3) The environment transitions to its next state according to the transition function $\mathbf{S}_{t+1} \sim T(\mathbf{S}_t, \mathbf{A}_{t+1}, \mathbf{D}_{t+1})$ (Algorithm 1).

(4) The attacker and defender are assessed rewards (and/or costs) for the time step.

When an active node is disabled by the defender, that node becomes inactive. If a node is activated by the attacker at the same step it is being disabled by the defender, the node remains inactive.

3.3 Payoff function

We denote by $\Omega_T = \{(\mathbf{A}_0, \mathbf{D}_0, \mathbf{S}_0), \ldots, (\mathbf{A}_T, \mathbf{D}_T, \mathbf{S}_T)\}$ the game history, which consists of all actions and resulting graph states at each time step. At time t, $\mathbf{S}_t$ is a resulting graph state when the attacker plays $\mathbf{A}_t$, the defender plays $\mathbf{D}_t$ and the previous graph state is $\mathbf{S}_{t-1}$. The defender and attacker's payoffs with respect to Ω_T, which comprise goal rewards and action costs, are computed as follows:

$$U^d(\Omega_T) = \sum_{t=1}^{T} \gamma^{t-1} \left[\sum_{v \in \mathbf{D}_t} c^d(v) + \sum_{v \in \mathbf{V}^g \cap \mathbf{S}_t} r^d(v) \right]$$

$$U^a(\Omega_T) = \sum_{t=1}^{T} \gamma^{t-1} \left[\sum_{e \in \mathbf{A}_t \cap \mathbf{E}^\vee} c^a(e) + \sum_{v \in \mathbf{A}_t \cap \mathbf{V}^\wedge} c^a(v) + \sum_{v \in \mathbf{V}^g \cap \mathbf{S}_t} r^a(v) \right].$$

Both players aim to maximize expected utility with respect to the distribution of Ω_T. Since the game is too complex for analytic solution, we propose heuristic strategies for both players and employ the simulation-based methodology EGTA to evaluate these strategies.

4 ATTACKER STRATEGIES

4.1 Attack candidate set

At $t + 1$, based on $\mathbf{S}_t$, the attacker needs to consider only $\vee$-exploits in $\mathbf{E}^\vee$ and $\wedge$-nodes in $\mathbf{V}^\wedge$ that can change the graph state at $t + 1$. We call this set of $\vee$-exploits and $\wedge$-nodes the *attack candidate set* at time $t + 1$, denoted by $\mathbf{\Psi}^a(\mathbf{S}_t)$ and defined as follows:

$$\{(u, v) \in \mathbf{E}^\vee \mid u \in \mathbf{S}_t, v \notin \mathbf{S}_t\} \cup \{v \in \mathbf{V}^\wedge \setminus \mathbf{S}_t \mid \pi^-(v) \subseteq \mathbf{S}_t\}$$

Essentially, $\mathbf{\Psi}^a(\mathbf{S}_t)$ consists of (i) $\vee$-exploits from active preconditions to inactive $\vee$-postconditions, and (ii) inactive $\wedge$-nodes for which all preconditions are active. Based on $\mathbf{\Psi}^a(\mathbf{S}_t)$, we propose a series of heuristic attack strategies, of increasing complexity.

4.2 Uniform attack strategy

Under the uniform attack strategy, the attacker chooses a fixed fraction of the candidate set $\mathbf{\Psi}^a(\mathbf{S}_t)$, uniformly at random.

4.3 Value-propagation attack strategy

The value-propagation strategy chooses actions based on a quantitative assessment of the candidates $\mathbf{\Psi}^a(\mathbf{S}_t)$. Intuitively, the value of an attack represents its impact on activating the goal nodes by the final time step T. The main idea of this strategy is to propagate the attacker rewards $r^a(w) > 0$ at inactive goal nodes $w \in \mathbf{V}^g \setminus \mathbf{S}_t$ backward to other nodes. The cost of attacking and the activation probabilities are incorporated accordingly. In the propagation process, there are multiple paths from goal nodes to each node. The attack value of a node is computed as the maximum value among

Algorithm 2: Compute Attack Value

1 Input: $t+1$, $\mathbf{S}_t$, and inverse topological order of $\mathbf{G}$, *itopo*($\mathbf{G}$);
2 Initialize node values $r^w(v,t') = 0$ and $r^w(w,t+1) = r^a(w)$ for all inactive nodes $v \in \mathbf{V} \setminus (\{w\} \cup \mathbf{S}_t)$, inactive goal nodes $w \in \mathbf{V}^g \setminus \mathbf{S}_t$, and time step $t' \geq t+1$;
3 **for** $u \in$ *itopo*($\mathbf{G}$) $\setminus \mathbf{S}_t$ **do**
4 **for** $v \in \pi^+(u) \setminus \mathbf{S}_t$ **do**
5 **for** $w \in \mathbf{V}^g \setminus (\mathbf{S}_t \cup \{u\}), t' = t+1, \ldots, T-1$ **do**
6 **if** $v \in \mathbf{V}^\wedge$ **then**
7 $r^w(v \to u, t'+1) = \frac{c^a(v) + p(v) r^w(v,t')}{|\pi^-(v) \setminus \mathbf{S}_t|^\alpha}$;
8 **else**
9 $r^w(v \to u, t'+1) = c^a(u,v) + p(u,v) r^w(v,t')$;
10 **if** $r^w(u,t'+1) < \gamma r^w(v \to u, t'+1)$ **then**
11 Update $r^w(u,t'+1) = \gamma r^w(v \to u, t'+1)$;
12 Return $\hat{r}(u) = \max_{w \in \mathbf{V}^g \setminus \mathbf{S}_t} \max_{t' \in \{t+1,\ldots,T\}} r^w(u,t')$, $\forall u \in \mathbf{V} \setminus \mathbf{S}_t$;

Algorithm 3: Random Activation

1 Input: $t+1$, $\mathbf{S}_t$, and topological order of $\mathbf{G}$, *topo*($\mathbf{G}$);
2 Initialize $p^{act}(v) = 1.0$, $t^{act}(v) = t$, and $pre(v) = \emptyset$, for all active nodes $v \in \mathbf{S}_t$;
3 Initialize $p^{act}(v) = p(v)$, $t^{act} = t+1$, and $pre(v) = \emptyset$ for all inactive root nodes $v \in \mathbf{V}^r \setminus \mathbf{S}_t$;
4 **for** $v \in$ *topo*($\mathbf{G}$) $\setminus \mathbf{S}_t$ **do**
5 **if** $v \in \mathbf{V}^\vee$ **then**
6 Randomly choose a precondition u to activate v with probability $p^{ra}(u,v) \propto p^{act}(u) \times p(u,v), \forall u \in \pi^-(v)$;
7 Update $p^{act}(v) = p^{act}(u) \times p(u,v)$;
8 Update $t^{act}(v) = t^{act}(u) + 1$;
9 Update $pre(v) = \{u\}$;
10 **else**
11 Update $p^{act}(v)$ with respect to all preconditions $\pi^-(v)$;
12 Update $t^{act}(v) = \max_{u \in \pi^-(v)} t^{act}(u) + 1$;
13 Update $pre(v) = \pi^-(v)$;
14 Return $\{(p^{act}(v), t^{act}(v), pre(v))\}$;

propagation paths reaching that node. This propagation process is illustrated in Algorithm 2.

Algorithm 2 leverages the directed acyclic topological structure of the attack graph to perform the goal-value propagation. We sort nodes according to the graph's topological order and start the propagation from leaf nodes following the inverse direction of the topological order. By doing so, we ensure that when a node is examined in the propagation process, all postconditions of that node have already been examined. As a result, we need to examine each node only once during the whole propagation process. In Algorithm 2, line 1 specifies input of the algorithm which includes the current time step $t+1$, the graph state in previous time step $\mathbf{S}_t$, and the inverse topological order of the graph $\mathbf{G}$, *itopo*($\mathbf{G}$). Line 2 initializes attack values $r^w(v,t')$ of inactive nodes v with respect to each inactive goal node w and time step t'. Intuitively, $r^w(v,t')$ indicates the attack value of node v with respect to propagation paths of length $t'-t-1$ from the inactive goal node $w \in \mathbf{V}^g \setminus \mathbf{S}_t$ to v. Given the time horizon $\mathbf{T} = \{0,\ldots,T\}$, we consider only paths of length up to $T-t-1$. At each iteration of evaluating a particular inactive node u, Algorithm 2 examines all inactive postconditions v of u and estimates the attack value propagated from v to u, $r^w(v \to u, t'+1)$. If node v is of $\wedge$-type, the attack value with respect to v, $c^a(v) + p(v) r^w(v,t')$ is equally distributed to all of its inactive preconditions including u (line 7). The propagation parameter α regulates the amount of distributed value. When $\alpha = 1.0$, in particular, that value is equally divided among these inactive preconditions. If node v is of $\vee$-type, u receives the attack value of $c^a(u,v) + p(u,v) r^w(v,t')$ from v (line 9). Since there are multiple propagation paths reaching node u, Algorithm 2 keeps the maximum propagated value (line 11). Finally, the attack value $\hat{r}(u)$ of each inactive node u is computed as the maximum over inactive goal nodes and time steps (line 12).

Based on attack values of inactive nodes, we compute the value of attacking candidates in $\mathbf{\Psi}^a(\mathbf{S}_t)$, incorporating the cost of each candidate and the corresponding activation probability as follows:

$$r(e) = \gamma^t \left[c^a(e) + p(e) \hat{r}(u) \right], \forall e = (v,u) \in \mathbf{\Psi}^a(\mathbf{S}_t)$$
$$r(u) = \gamma^t \left[c^a(u) + p(u) \hat{r}(u) \right], \forall u \in \mathbf{\Psi}^a(\mathbf{S}_t),$$

Finally, the value-propagation attack strategy selects attacks to execute probabilistically, based on the assessed attack value. The strategy first determines the number of attacks to execute, then selects that number of attacks using a conditional logistic function. Specifically, the probability that exploit $e \in \mathbf{\Psi}^a(\mathbf{S}_t)$ is selected is as follows:

$$P(e) = \frac{\exp\left[\eta^a r(e)\right]}{\sum\limits_{e' \in \mathbf{\Psi}^a(\mathbf{S}_t) \cap \mathbf{E}^\vee} \exp\left[\eta^a r(e')\right] + \sum\limits_{u \in \mathbf{\Psi}^a(\mathbf{S}_t) \cap \mathbf{V}^\wedge} \exp\left[\eta^a r(u)\right]}.$$

The probability for a $\wedge$-node in $\mathbf{\Psi}^a(\mathbf{S}_t)$ is defined similarly. Model parameter $\eta^a \in [0, +\infty)$ governs how strictly the choice follows assessed attack values.

4.4 Sampled-activation attack strategy

Like the value-propagation strategy, the sampled-activation attack strategy selects actions based on a quantitative assessment of relative value. Rather than propagation backward from goal nodes, this strategy constructs estimates by forward sampling from the current candidates $\mathbf{\Psi}^a(\mathbf{S}_t)$.

4.4.1 Random activation process. The attacker uses the topological order of the attack graph to sample paths of activation from the current graph state $\mathbf{S}_t$. Following this order ensures that all preconditions are visited before any corresponding postconditions. For each visited node v, we keep track of a set of preconditions $pre(v)$ which are selected in the sampled-activation process to activate v. If v is a $\wedge$-node, the set $pre(v)$ consists of all preconditions of v. If v is a $\vee$-node, we randomly select a precondition $pre(v)$ to use to activate v. This randomized selection is explained below. Each inactive node v is assigned an activation probability $p^{act}(v)$ and an activation time step $t^{act}(v)$ according to the random action the

attacker takes. The activation probability $p^{act}(v)$ and the activation time step $t^{act}(v)$ represent the probability and the time step node v will become active if the attacker follows the sampled action sequence to activate v (and the defender takes no action).

The random activation process is illustrated in Algorithm 3. When visiting an inactive $\vee$-node $v \in \mathbf{V}^{\vee} \setminus \mathbf{S}_t$, the attacker randomly chooses a precondition $u \in \pi^{-}(v)$ (from which to activate that $\vee$-node) with a probability $p^{ra}(u, v)$. This probability is computed based on activation probability $p(u, v)$ and activation probability $p^{act}(u)$ of the associated precondition u (line 6). Intuitively, $p^{act}(u) \times p(u, v)$ is the probability v becomes active if the attacker chooses the exploit (u, v) to activate v in the random activation. Accordingly, the higher $p^{act}(u) \times p(u, v)$ is, the higher the chance that u is the selected precondition for v. We then update v with respect to the selected u (lines 7–9). When visiting an inactive $\wedge$-node v, all preconditions of v are required to activate v (line 13). Thus, the activation time step of v must be computed based on the maximum activation time step of v's preconditions (line 12).

The activation probability $p^{act}(v)$ of the inactive $\wedge$-node v involves the activation probability $p^{act}(u)$ of all of its preconditions $u \in \pi^{-}(v)$. These activation probabilities $\{p^{act}(u) \mid u \in \pi^{-}(v)\}$ depend on the sequences of nodes (which may not be disjoint) chosen in the random activation process to activate v's preconditions. Therefore, we need to backtrack over all nodes in the activation process of v to compute $p^{act}(v)$. We denote this sequence of nodes as $seq(v) = seq^{\vee}(v) \cup seq^{\wedge}(v)$ where $seq^{\vee}(v)$ consists of $\vee$-nodes only and $seq^{\wedge}(v)$ consists of $\wedge$-nodes.

$$seq(v) = \{v\} \cup pre(v) \cup pre(pre(v)) \ldots$$

Based on $seq(v)$, the activation probability, $p^{act}(v)$, is computed as follows, which comprises the activation probabilities of all edges and nodes involved in activating v:

$$p^{act}(v) = \left[\prod\nolimits_{u \in seq^{\vee}(v)} p(pre(u), u)\right]\left[\prod\nolimits_{u \in seq^{\wedge}(v)} p(u)\right].$$

4.4.2 Greedy attack strategy. At the end of a random activation, we obtain a sequence of nodes chosen to activate each inactive goal node. Thus, we can estimate the attack value of each subset $\hat{\mathbf{V}}^g \subseteq \mathbf{V}^g \setminus \mathbf{S}_t$ of inactive goal nodes according to the random activation:

$$\begin{aligned} r(\hat{\mathbf{V}}^g) = &\sum\nolimits_{v \in \hat{\mathbf{V}}^g} p^{act}(v) r^a(v) \gamma^{t^{act}(v)-1} \\ &+ \sum\nolimits_{v \in seq^{\wedge}(\hat{\mathbf{V}}^g)} \frac{p^{act}(v)}{p(v)} c^a(v) \gamma^{t^{act}(v)-1} \\ &+ \sum\nolimits_{v \in seq^{\vee}(\hat{\mathbf{V}}^g)} \frac{p^{act}(v)}{p(pre(v), v)} c^a(pre(v), v) \gamma^{t^{act}(v)-1} \end{aligned}$$

where the first term accounts for the rewards of the goal nodes in the subset, and the second and third terms account for the costs of activating inactive nodes in the corresponding sampled-activation sequences. In particular, $seq^{\wedge}(\hat{\mathbf{V}}^g) = \cup_{v \in \hat{\mathbf{V}}^g} seq^{\wedge}(v)$ and $seq^{\vee}(\hat{\mathbf{V}}^g) = \cup_{v \in \hat{\mathbf{V}}^g} seq^{\vee}(v)$. The probability $\frac{p^{act}(v)}{p(v)}$ indicates the probability all preconditions of the $\wedge$-node v become active and thus the attacker can activate v with a cost $c^a(v)$. Similarly, $\frac{p^{act}(v)}{p(pre(v), v)}$ is the probability that the chosen precondition of the $\vee$-node v becomes active and thus the attacker can activate v with a cost $c^a(pre(v), v)$.

The sampled-activation attack strategy aims to find a subset of inactive goal nodes to activate following the random activation which maximizes the attack value. However, finding an optimal subset of inactive goal nodes is computationally expensive, because there is an exponential number of subsets of inactive goal nodes to consider. Therefore, we use the greedy approach to find a reasonable subset of inactive goal nodes to attempt to activate. Essentially, given the current subset of selected inactive goal nodes $\hat{\mathbf{V}}^g$ (which was initially empty), we iteratively find the next best inactive goal node $u \in \mathbf{V}^g \setminus \mathbf{S}_t$ such that $r(\hat{\mathbf{V}}^g \cup \{u\})$ is maximized and add u to $\hat{\mathbf{V}}^g$. This greedy process continues until the attack value stops increasing: $r(\hat{\mathbf{V}}^g \cup \{u\}) - r(\hat{\mathbf{V}}^g) \leq 0$. Based on the subset of chosen goal nodes, we obtain a corresponding candidate subset:

$$\begin{aligned} &\{v \mid v \in seq^{\wedge}(\hat{\mathbf{V}}^g), pre(v) \subseteq \mathbf{S}_t\} \\ &\qquad \cup \{(u, v) \mid v \in seq^{\vee}(\hat{\mathbf{V}}^g), pre(v) = \{u\}, u \in \mathbf{S}_t\} \end{aligned}$$

which need to activate in current time step $t+1$ according to the sampled-activation process in order to activate the goal subset $\hat{\mathbf{V}}^g$ subsequently. We assign the value of the goal subset to this candidate subset. By running random activation multiple times, we obtain a set of candidate subsets. Finally, the attacker action at $t+1$ is randomly chosen among these subsets of candidates following a conditional logistic distribution with respect to the attack values of these subsets.

5 DEFENDER STRATEGIES

At each time step, since the defender does not know the true graph states, it is important for her to reason about possible graph states before choosing defense actions. In the following, we first study the defender's belief update on graph states at each time step and then propose different defense heuristic strategies.

5.1 Defender belief update

As mentioned before, in our game, the defender knows the initial graph state, in which all nodes are inactive. As the game evolves, the defender needs to take into account both her observations and the assumed attacker strategy to update her belief on possible graph states. We denote by $\mathbf{b}_t = \{b_t(\mathbf{S}_t)\}$ the defender's belief at the end of time step t, where $b_t(\mathbf{S}_t)$ is the probability the graph state at time step t is $\mathbf{S}_t$, and $\sum_{\mathbf{S}_t} b_t(\mathbf{S}_t) = 1.0$. At time step 0, $b_0(\emptyset) = 1.0$. Based on the defender's belief $\mathbf{b}_{t-1}$, action $\mathbf{D}_t$, and observation $\mathbf{O}_t$, we can update the defender's belief $\mathbf{b}_t$ as follows:

$$\begin{aligned} &b_t(\mathbf{S}_t) = p(\mathbf{S}_t \mid \mathbf{b}_{t-1}, \mathbf{D}_t, \mathbf{O}_t) \propto p(\mathbf{S}_t, \mathbf{b}_{t-1}, \mathbf{D}_t, \mathbf{O}_t) \qquad (1) \\ &\propto p(\mathbf{O}_t \mid \mathbf{S}_t) \times \\ &\left[\sum_{\mathbf{S}_{t-1} \in \Im(\mathbf{b}_{t-1})} b_{t-1}(\mathbf{S}_{t-1}) \sum_{\mathbf{A}_t \in \mathfrak{A}(\mathbf{S}_{t-1})} p(\mathbf{S}_t \mid \mathbf{A}_t, \mathbf{D}_t, \mathbf{S}_{t-1}) p(\mathbf{A}_t \mid \mathbf{S}_{t-1})\right] \end{aligned}$$

where $p(\mathbf{O}_t \mid \mathbf{S}_t)$ is the probability that the defender receives observation $\mathbf{O}_t$ given the graph state is $\mathbf{S}_t$ at time step t. Because the alerts with respect to each node are independent from other nodes, we can compute this observation probability as follows:

$$p(\mathbf{O}_t \mid \mathbf{S}_t) = \prod\nolimits_{v \in \mathbf{V}} p\left(o_t(v) \mid s_t(v)\right).$$

In addition, $\Im(\mathbf{b}_{t-1})$ is the belief state set associated with the defender's belief $\mathbf{b}_{t-1}$ at time step $t-1$: $\Im(\mathbf{b}_{t-1}) = \{\mathbf{S}_{t-1} \mid b_{t-1}(\mathbf{S}_{t-1}) >$

0}. The probability of state transition $p(\mathrm{S}_t \mid \mathrm{A}_t, \mathrm{D}_t, \mathrm{S}_{t-1})$ is computed based on the state transition of every node:

$$p(\mathrm{S}_t \mid \mathrm{A}_t, \mathrm{D}_t, \mathrm{S}_{t-1}) = \prod\nolimits_v p\left(s_t(v) \mid \mathrm{A}_t, \mathrm{D}_t, \mathrm{S}_{t-1}\right),$$

in which the transition probability, $p\left(s_t(v) \mid \mathrm{A}_t, \mathrm{D}_t, \mathrm{S}_{t-1}\right)$, is computed in Algorithm 1. Finally, the set $\mathfrak{A}(\mathrm{S}_{t-1})$ consists of all possible attack actions with respect to the graph state S_{t-1}. The probability $p(\mathrm{A}_t \mid \mathrm{S}_{t-1})$ is the probability the attacker takes action A_t at time step t given the graph state at the end of time step $t-1$ is S_{t-1}.

Exactly computing the defender's belief (Equation 1) over all possible graph states at each time step is impractical. Indeed, there are exponentially many graph states to explore, as well as an exponential number of possible attack actions. To overcome this computational challenge, we apply particle filtering [20], a Monte Carlo sampling method for performing state inference, given noisy observations at each time step. This approach allows us to limit the number of graph states and attack actions considered.

5.2 Simple defense strategies

We present four simple defense strategies which do not take into account the defender belief on possible graph states. Each targets a specific group of nodes to disable at each time step $t+1$.

Uniform strategy. The defender chooses nodes in the graph to disable uniformly at random. The number of nodes chosen is a certain fraction of $|\mathbf{V}|$.

Min-cut uniform strategy. The defender chooses nodes in the min-cut set of the graph to disable uniformly at random. The min-cut set is the minimum set of edges that removing them disconnects the root nodes from the goal nodes. The number of chosen nodes is a certain fraction of the number of nodes in the min-cut set.

Root-node uniform strategy. The defender chooses root nodes to disable uniformly at random. The number of nodes chosen is a certain fraction of $|\mathbf{V}^r|$.

Goal-node strategy. The defender randomly chooses goal nodes to disable with probabilities depending on the rewards and costs associated with these goal nodes. The probability of disabling each $v \in \mathbf{V}^g$ is based on the conditional logistic function:

$$p(v \mid t+1) = \frac{\exp[\eta^d \gamma^t(-r^d(v) + c^d(v))]}{\sum_{u \in \mathbf{V}^g} \exp[\eta^d \gamma^t(-r^d(u) + c^d(u))]},$$

where $\gamma^t(-r^d(v) + c^d(v))$ indicates the potential value the defender receives for disabling v. In addition, η^d is the parameter of the logistic function which is predetermined. The number of nodes chosen will be a certain fraction of the number of goal nodes. Then we draw that many nodes from the distribution.

In the following section, we propose two new defense strategies that take into account the defender's belief on possible graph states at each time step, called the *value-propagation* and *sampled-activation* defense strategies. These two defense strategies use concepts similar to the value-propagation and sampled-activation attack strategies.

5.3 Defense candidate set

At each time step $t+1$, the defender has belief $\mathbf{b}_t$ on possible graph states at the end of time step t. For each state in the belief set $\mathrm{S}_t \in \mathfrak{S}(\mathbf{b}_t)$, we define the defense candidate set, $\mathbf{\Psi}^d(\mathrm{S}_t)$, which consists of: active goal nodes, $\wedge$-nodes and $\vee$-postconditions of exploits in the attack candidate set $\mathbf{\Psi}^a(\mathrm{S}_t)$. We aim at disabling not only active goal nodes but also nodes in the $\mathbf{\Psi}^a(\mathrm{S}_t)$, to prevent the attacker from intruding further.

$$\mathbf{\Psi}^d(\mathrm{S}_t) = (\mathbf{V}^g \cap \mathrm{S}_t) \cup \{\mathbf{\Psi}^a(\mathrm{S}_t) \cap \mathbf{V}^\wedge\} \cup \mathrm{post}(\mathbf{\Psi}^a(\mathrm{S}_t)),$$

where $\mathrm{post}(\mathbf{\Psi}^a(\mathrm{S}_t))$ consists of $\vee$-postconditions of exploits in $\mathbf{\Psi}^a(\mathrm{S}_t)$.

5.4 Value-propagation defense strategy

For each possible graph state $\mathrm{S}_t \in \mathfrak{S}(\mathbf{b}_t)$, we first estimate the propagated defense reward $\hat{r}(u \mid \mathrm{S}_t)$ of each node $u \in \mathbf{V}$ by propagating the defender's rewards $r^d(w) < 0$ at inactive goal nodes $w \in \mathbf{V}^g \setminus \mathrm{S}_t$ to u. Intuitively, the propagated defense reward associated with each node accounts for the potential loss the defender can prevent for blocking that node. The detail is presented in Algorithm 5 in the appendix A, with the idea is similar to the value-propagation attack strategy. Based on that, we then can estimate the defense values, $r(u \mid \mathrm{S}_t)$, for defense candidate nodes $u \in \mathbf{\Psi}^d(\mathrm{S}_t)$ as follows:

$$r(u \mid \mathrm{S}_t) = c^d(u) + \begin{cases} -\hat{r}(u \mid \mathrm{S}_t) - r^d(u), \text{ if } u \in \mathrm{S}_t \cap \mathbf{V}^g \\ -\frac{\sum_k p\left(s_{t+1}(u)=1 \mid \mathrm{A}^k, \mathrm{S}_t\right)}{N^a} \hat{r}(u \mid \mathrm{S}_t), \text{ otherwise} \end{cases}$$

For active goal nodes $u \in \mathbf{V}^g \cap \mathrm{S}_t$, $r(u \mid \mathrm{S}_t)$ comprises not only the cost $c^d(u)$ but also the propagated defense reward $\hat{r}(u \mid \mathrm{S}_t)$ and the defender's reward, $r^d(u)$, at u. For other defense candidate nodes, $r(u \mid \mathrm{S}_t)$ takes into account the attack strategy to compute the probability u becomes active as a result of the attacker's action at $t+1$. Essentially, the higher probability a candidate node u becomes active, the higher defense value for the defender to disable that node. Because there is an exponential number of possible attack actions, we sample a set of N^a attack actions, denoted by $\{\mathrm{A}^k\}$ where $k = 1, 2, \ldots, N^a$, according to the assumed attack strategy. The defense value for each u in $\mathbf{\Psi}^d(\mathrm{S}_t) \setminus (\mathbf{V}^g \cap \mathrm{S}_t)$ takes into account the probability u becomes active (as a result of sampled attack actions), denoted by $p\left(s_{t+1}(u) = 1 \mid \mathrm{A}^k, \mathrm{S}_t\right)$, which is equal to:

$$\begin{cases} I(u \in post(\mathrm{A}^k)) \left[1 - \prod\limits_{e \in \mathrm{A}^k \mid \mathrm{post}(e) = u} (1 - p(e))\right], \text{ if } u \in \mathrm{post}(\mathbf{\Psi}^a(\mathrm{S}_t)) \\ I(u \in \mathrm{A}^k) p(u), \text{ if } u \in \mathbf{\Psi}^a(\mathrm{S}_t) \cap \mathbf{V}^\wedge, \end{cases}$$

where $I(\Phi)$ is a binary indicator for condition Φ. Finally, based on the defense values with respect to each graph state in the belief state set, $\mathfrak{S}(\mathbf{b}_t)$, we can compute the expected defense values over $\mathfrak{S}(\mathbf{b}_t)$ as follows:

$$\bar{r}(u) = \gamma^t \left[\sum\nolimits_{\mathrm{S}_t \in \mathfrak{S}(\mathbf{b}_t)} b_t(\mathrm{S}_t) r(u \mid \mathrm{S}_t)\right],$$

for all defense candidate nodes $u \in \cup_{\mathrm{S}_t} \mathbf{\Psi}^d(\mathrm{S}_t)$. The probability the defender will choose each node u to disable is computed according to the conditional logistic function with a predetermined parameter value. The number of chosen nodes to disable is a certain fraction of the cardinality of the defense candidate set $\cup_{\mathrm{S}_t} \mathbf{\Psi}^d(\mathrm{S}_t)$.

5.5 Sampled-activation defense strategy

In this defense strategy, we leverage the random activation process as described in Section 4.4 to reason about potential attack paths the attacker may follow to attack goal nodes. Based on this reasoning, we can estimate the defense value for each possible defense action and then select the action which leads to the highest defense value.

5.5.1 Sample attack plans. At time step $t+1$, for each possible graph state $\mathbf{S}_t \in \mathfrak{S}(\mathbf{b}_t)$, we sample N^r random activations, each resulting in sampled-activation sequences toward inactive goal nodes. We also sample a set of N^a attack actions, $\{\mathbf{A}^k\}$ where $k = 1, 2, \ldots, N^a$, according to the defender's assumption about the attacker's strategy. For each $\mathbf{A}^k$, we select the best random activation among the N^r sampled-activation samples such that performing $\mathbf{A}^k$ at $t+1$ can lead to the activation the subset of inactive goal nodes with the highest attack value according to that random activation (Section 4.4). We denote by $ra(\mathbf{A}^k)$ the sequence of nodes (sorted according to the topological order of the graph) which can be activated in future time steps based on $\mathbf{A}^k$ and the corresponding selected random activation. We call each pair $(\mathbf{A}^k, ra(\mathbf{A}^k))$, an attack plan for the attacker.

5.5.2 Estimate defense values. Based on sampled attack plans $(\mathbf{A}^k, ra(\mathbf{A}^k))$, $\forall k = 1, 2, \ldots, N^a$, we can estimate the defense value of any defense action $\mathbf{D}_{t+1} \subseteq \mathbf{V}$ with respect to $\mathbf{S}_t$ as follows:

$$r(\mathbf{D}_{t+1} \mid \mathbf{S}_t) = \frac{\sum_k r(\mathbf{D}_{t+1} \mid \mathbf{S}_t, \mathbf{A}^k, ra(\mathbf{A}^k))}{N^a},$$

where $r(\mathbf{D}_{t+1} \mid \mathbf{S}_t, \mathbf{A}^k, ra(\mathbf{A}^k))$ is the defender's value for playing action $\mathbf{D}_{t+1}$ against $(\mathbf{A}^k, ra(\mathbf{A}^k))$ which is determined based on which goal nodes can potentially become active given players' actions $\mathbf{D}_{t+1}$ and $(\mathbf{A}^k, ra(\mathbf{A}^k))$. To determine these goal nodes, we iteratively examine nodes in $ra(\mathbf{A}^k)$ to find which goal nodes $v \in \mathbf{V}^g$ of which sequence $seq(v)$ is not blocked by the defender's action $\mathbf{D}_{t+1}$. Recall that $seq(v)$ is a sequence of nodes to activate in the chosen random activation to activate v.

This search process is shown in Algorithm 4, where $isBlocked(u)$ indicates if the attack sequence to node u according to $(\mathbf{A}^k, ra(\mathbf{A}^k))$ is blocked by the defender ($isBlocked(u) = 1$) or not ($isBlocked(u) = 0$). Initially, $isBlocked(v) = 0$ for all nodes v in $ra(\mathbf{A}^k) \setminus \mathbf{D}_{t+1}$ while $isBlocked(v) = 1$ for all $v \in \mathbf{D}_{t+1}$. While examining non-root nodes in $ra(\mathbf{A}^k)$, an $\vee$-node u is updated to $isBlocked(u) = 1$ if all preconditions of u in $pre(u)$ are blocked. On the other hand, an $\wedge$-node u becomes blocked when any of its preconditions is blocked. Given $\{isBlocked(v)\}$, we can estimate the defense value $r(\mathbf{D}_{t+1} \mid \mathbf{S}_t, \mathbf{A}^k, ra(\mathbf{A}^k))$ at time step $t+1$ as follows:

$$r(\mathbf{D}_{t+1} \mid \mathbf{S}_t, \mathbf{A}^k, ra(\mathbf{A}^k)) = \sum_{v \in \mathbf{D}_{t+1}} c^d(v)\gamma^t + \sum_{v \in \mathbf{V}^g} p^{act}(v) r^d(v) \gamma^{t^{act}(v)-1} (1 - isBlocked(v))$$

where the first term is the cost of performing $\mathbf{D}_{t+1}$. The second term accounts for the potential loss of the defender which comprises the blocked status, the activation probability, the activation time step, and the rewards of all the goal nodes. Finally, the expected defense value of each $\mathbf{D}_{t+1}$ over the defender's belief is computed as follows:

$$r(\mathbf{D}_{t+1} \mid \mathbf{b}_t) = \sum_{\mathbf{S}_t} b_t(\mathbf{S}_t) r(\mathbf{D}_{t+1} \mid \mathbf{S}_t)$$

Algorithm 4: Find blocked nodes

```
Input: S_t, D_{t+1}, A^k, ra(A^k);
Initialize block status isBlocked(v) = 0 for all
  v ∈ ra(A^k) \ D_{t+1} and isBlocked(v) = 1 for all v ∈ D_{t+1} ;
for u ∈ ra(A^k) \ V^r with isBlocked(u) = 0 do
    if u ∈ V^∨ then
        if isBlocked(pre(u)) then
            isBlocked(u) = 1;
    else
        if isBlocked(v) for some v ∈ pre(u) then
            isBlocked(u) = 1;
Return {isBlocked(u)}.
```

5.5.3 Greedy defense strategy. Finding an optimal defense action is computationally expensive, because there is an exponential number of possible defense actions. Therefore, we propose two different greedy heuristics to overcome this computational challenge.

Static greedy heuristic. This heuristic greedily finds a reasonable set of nodes to disable over the defender's belief $\mathbf{b}_t$. Given the current set of selected nodes $\mathbf{D}_{t+1}$ (which was initially empty), the heuristic finds the next best node u such that $r(\mathbf{D}_{t+1} \cup \{u\} \mid \mathbf{b}_t)$ is maximized. The iteration process stops when disabling new nodes does not increase the defender's value, i.e., $r(\mathbf{D}_{t+1} \cup \{u\} \mid \mathbf{b}_t) - r(\mathbf{D}_{t+1} \mid \mathbf{b}_t) \leq 0$ for all u.

Randomized greedy heuristic. This heuristic greedily finds a reasonable set of nodes $\mathbf{D}_{t+1}(\mathbf{S}_t)$ to disable with respect to each $\mathbf{S}_t$ in $\mathfrak{S}(\mathbf{b}_t)$. For each $\mathbf{S}_t \in \mathfrak{S}(\mathbf{b}_t)$, given the current set of selected nodes $\mathbf{D}_{t+1}(\mathbf{S}_t)$ (which was initially empty), the heuristic finds the next best node u such that $r(\mathbf{D}_{t+1}(\mathbf{S}_t) \mid \mathbf{S}_t)$ is maximized. As a result, we obtain multiple greedy defense actions $\{\mathbf{D}_{t+1}(\mathbf{S}_t)\}$ corresponding to possible game states $\mathbf{S}_t$ in $\mathfrak{S}(\mathbf{b}_t)$. The defender then randomizes her choice over $\{\mathbf{D}_{t+1}(\mathbf{S}_t)\}$ according to a conditional logistic distribution with respect to the defense value $\{r(\mathbf{D}_{t+1}(\mathbf{S}_t) \mid \mathbf{b}_t)\}$.

6 EXPERIMENTS

We evaluate the solution quality of the proposed strategies in various game settings with different graph topologies, node type ratios, and the defender's observation noise levels.

6.1 Player strategies

We tune the strategies for the players by adjusting their parameter values. In our experiments, the percentage of candidates chosen to attack for the attacker's strategies is $p^a \in \{0.3, 0.5\}$ of the total number of attack candidates. The logistic parameter value is $\eta^a \in \{1.0, 3.0\}$. As a result, our experiments consist of nine different *attack strategy instances*: (i) a No-op (*aNoop*) instance in which the attacker does not perform any attack action; (ii) two Uniform (*aUniform*) instances with $p^a \in \{0.3, 0.5\}$; (iii) four Value-propagation (*aVP*) instances with $p^a \in \{0.3, 0.5\}$ and $\eta^a \in \{1.0, 3.0\}$; and (iv) two Sampled-activation (*aSA*) instances with $\eta^a \in \{1.0, 3.0\}$.

The percentage of nodes chosen to protect for the defender strategies is $p^d \in \{0.3, 0.5\}$ of the total number of defense candidate nodes. The logistic parameter value is $\eta^d \in \{1.0, 3.0\}$. In addition, the defender's assumption about the attacker strategy considers the same set of aforementioned attack parameter values. Thus, we evaluate 43 different *defense strategy instances*: (i) a Noop (*dNoop*) instance in which the defender does not perform any defense action; (ii) two Uniform (*dUniform*), two Min-cut uniform (*dMincut*), and two Root-only uniform (*dRoot-only*) instances with $p^d \in \{0.3, 0.5\}$; (iii) four Goal-only (*dGoal-only*) instances with $p^d \in \{0.3, 0.5\}$ and $\eta^d \in \{1.0, 3.0\}$; (iv) 16 *aVP-dVP* instances — the defender follows the value-propagation defense strategy while assuming the attacker follows the value-propagation attack strategy. These 16 defense strategy instances correspond to: $p^d \in \{0.3, 0.5\}$ and $\eta^d \in \{1.0, 3.0\}$, $p^a \in \{0.3, 0.5\}$ and $\eta^a \in \{1.0, 3.0\}$; (v) eight *aSA-dSA* instances. These eight strategy instances correspond to $\eta^d \in \{1.0, 3.0\}$, $\eta^a \in \{1.0, 3.0\}$, and whether the defender uses randomized or static greedy heuristics; and finally (vi) eight *aVP-dSA* instances. These eight strategies correspond to $\eta^d \in \{1.0, 3.0\}$, $p^a \in \{0.3, 0.5\}$ and $\eta^a \in \{1.0, 3.0\}$.

6.2 Simulation settings

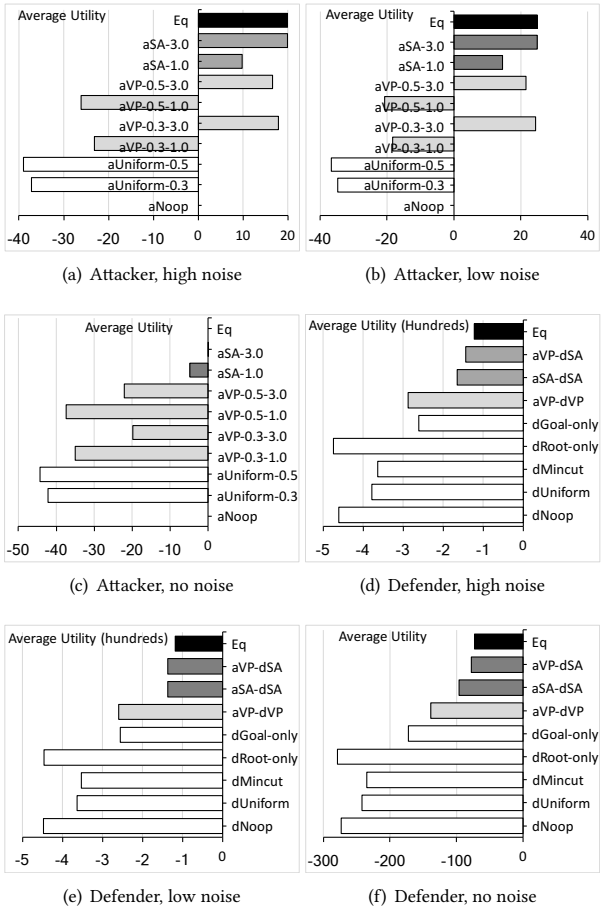
We consider two types of graph topologies: (i) layered directed acyclic graphs (layered DAGs); and (ii) random directed acyclic graphs (random DAGs). Graphs in the former case consist of multiple separate layers with edges connecting only nodes in consecutive layers. We generate 5-layered DAGs. The k^{th} layer ($k = 1, \ldots, 5$) has $25 \times 0.8^{k-1}$ nodes. All nodes in the last layer are goal nodes. In addition, edges are generated to connect every node at each layer to 50% of nodes at the next layer (which are chosen uniformly at random). In the latter case, random DAGs are generated with $|\mathbf{V}| = 100$ and $|\mathbf{E}| = 300$. In addition to leaf nodes, other nodes in random DAGs are selected as goal nodes uniformly at random given a fixed number of goal nodes (which is 15 in our experiments). The number of $\wedge$-nodes is either 0% or 50% of the nodes in graphs.

The defender's cost to disable each node $u \in \mathbf{V}$, $c^d(u)$, is generated between $1.2^{l^{min}(u)-1} \times [-1.0, -0.5]$ uniformly at random where $l^{min}(u)$ is the shortest distance from root nodes to u. The attacker's costs are generated similarly. The attacker reward and the defender's penalty at each goal node u are generated within $1.2^{l^{min}(u)-1} \times \{[10.0, 20.0], [-20.0, -10.0]\}$ uniformly at random. Finally, the activation probability associated with each edge with an $\vee$- postcondition and with each $\wedge$-node are randomly generated within $[0.6, 0.8]$ and $[0.8, 1.0]$ respectively.

We consider three cases of observation noise levels: (i) high noise — the signal probabilities $p(1_v \mid s(v) = 1)$ and $p(1_v \mid s(v) = 0)$ are generated within $[0.6, 0.8]$ and $[0.2, 0.4]$ uniformly at random respectively; (ii) low noise — $p(1_v \mid s(v) = 1)$ and $p(1_v \mid s(v) = 0)$ are generated within $[0.8, 1.0]$ and $[0.0, 0.2]$; and (iii) no noise — $p(1_v \mid s(v) = 1) = 1.0$ and $p(1_v \mid s(v) = 0) = 0.0$. The number of time steps is $T = 10$. The discount factor is $\gamma = 0.9$.

6.3 Strategy evaluation

Based on the aforementioned settings, we generated 10 different games in each of our experiments. For each game, we ran 500 simulations to estimate the payoff of each pair of players' strategy instances. As a result, we obtain a payoff matrix for each game based on which we can compute Nash equilibria using Gambit [14]. We compute the utility each player obtains for playing the proposed strategy instances (instead of the equilibrium strategy) against the opponent's equilibrium strategy. We compare that with the utility of the players in the equilibria to evaluate the solution quality of the proposed strategies. Each data point of our results is averaged over the 10 games. In addition, instead of showing results of every individual defense strategy instance (43 in total), we present results of the defense strategies averaged over all corresponding instances.



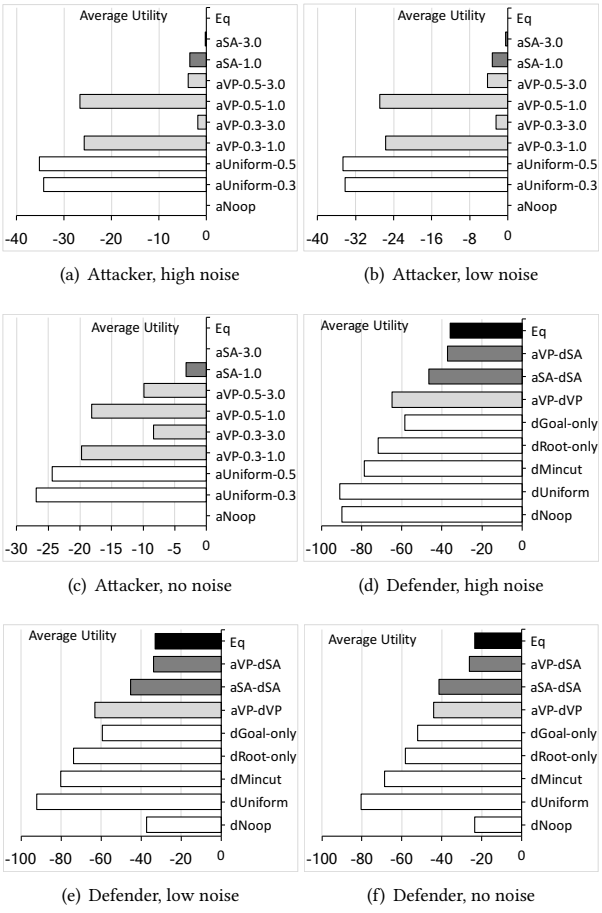
(a) Attacker, high noise (b) Attacker, low noise

(c) Attacker, no noise (d) Defender, high noise

(e) Defender, low noise (f) Defender, no noise

Figure 1: Strategy evaluation, layered DAGs, 0% $\wedge$-nodes

6.3.1 Results on layered DAGs. Our first set of experiments is based on layered DAGs, as shown in Figure 1 (0% $\wedge$-nodes) and Figure 2 (50% $\wedge$-nodes). In these figures, the x-axis represents the defender's or the attacker's expected utility, and the y-axis represents the corresponding strategies played by the players. For example, Figures 1(a)(b)(c) show the attacker's utilities for playing the strategies indicated on the y-axis against the defender's equilibrium strategy, when the percentage of $\wedge$-nodes in graphs is 0%.



(a) Attacker, high noise (b) Attacker, low noise

(c) Attacker, no noise (d) Defender, high noise

(e) Defender, low noise (f) Defender, no noise

Figure 2: Strategy evaluation, layered DAGs, 50% ∧-nodes

In Figures 1(d)(e)(f), and 2(d)(e)(f), the defense sampled-activation strategy (aVP-dSA and aSA-dSA) always obtains the defense utility close to the equilibrium utility (Eq) in all game settings regardless of the assumed attack strategies (i.e., whether the attacker follows the value-propagation or sampled-activation attack strategies with certain parameter values). In fact, when the percentage of ∧-nodes is 0%, the Nash equilibria obtained for all games based on EGTA only comprises the defender's sampled-activation strategy instances. This result shows that the sampled-activation defense strategy is robust to the defender's uncertainty about the attacker's strategy. Furthermore, when the observation noise level increases from no noise to high noise, the defender's sampled-activation strategies does not suffer from a significant loss in her utility. This result implies that the sampled-activation defense strategies are also robust to the defender's uncertainty about true graph states. Among the no-belief-update strategies, the goal-only strategy outperforms the root-only, min-cut, and uniform strategies in all game settings. This result shows that goal-only is a good candidate strategy when the defender's belief update is not taken into account. In addition, the goal-only strategy even obtains a higher utility than aVP-dVP in the cases of low and high observation noise.

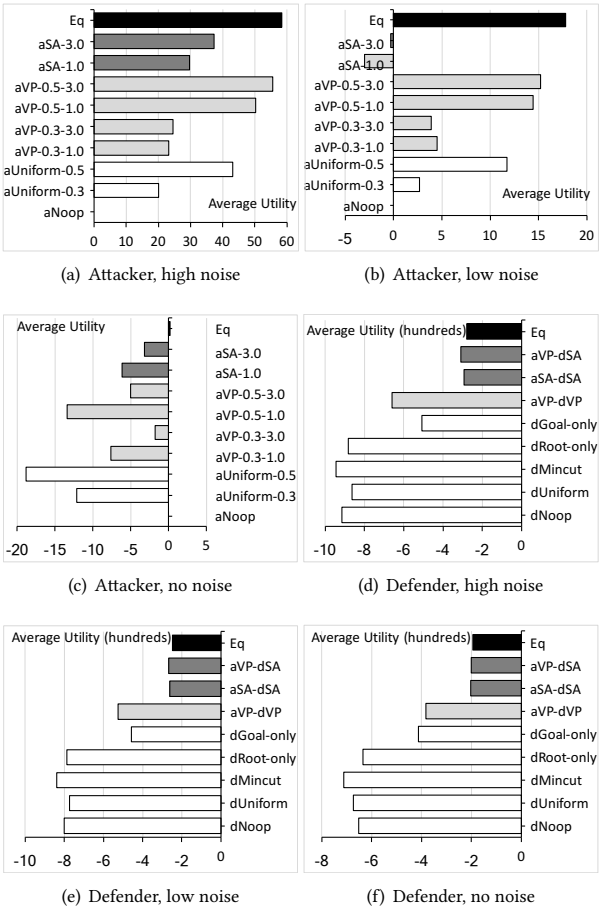
Figures 1(a)(b)(c) and 2(a)(b)(c) show that in all game settings, the attack sampled-activation strategy (i.e., aSA-3.0 and aSA-1.0) consistently obtains high attack utility compared with the attacker's equilibrium strategy (Eq). Even though the defender's equilibrium strategies focus on competing against the sampled-activation attack strategy, this attack strategy still performs well. The utility obtained by the attacker's value-propagation strategy (aVP-0.5-3.0, aVP-0.5-1.0, aVP-0.3-3.0, and aVP-0.3-1.0), on the other hand, varies depending on the value of the attack logistic parameter (η^a). In particular, both aVP-0.5-3.0 and aVP-0.3-3.0 with $\eta^a = 3.0$ obtain a considerably higher utility for the attacker compared with aVP-0.5-1.0 and aVP-0.3-1.0 with $\eta^a = 1.0$. Compared to all other strategies, the attacker's uniform strategy (aUniform-0.3 and aUniform-0.5) obtains the lowest attack utility.

When the percentage of ∧-nodes is 50%, aNoop gets a high probability in the attacker's equilibrium strategies in all game settings. In Figures 2(a)(b)(c), aNoop obtains the attacker's utility (which is zero) approximately the same as the equilibrium attack strategy. In fact, when the number of ∧-nodes is large, it is difficult for the attacker to intrude deeper in the attack graph, because compromising ∧-nodes takes more effort. Consequently, the defender obtains a significantly higher utility when the percentage of ∧-nodes is 50% (Figures 2(d)(e)(f)) than when it is 0% (Figures 1(d)(e)(f)). The dNoop strategy also involves in the defender's equilibrium strategy when the percentage is 50%. Finally, in Figure 1(c), the attacker's equilibrium utility is approximately zero even when all the nodes are of ∨-type. This result shows that when the defender knows graph states, our sampled-activation defense strategy is highly effective such that the attacker cannot achieve any benefit from attacking.

6.3.2 Results on random DAGs. Our second set of experiments are based on random DAGs (Figure 3). These figures show that the solution quality of the defender's strategies are similar to the case of layered DAGs. The defender's sampled-activation strategy obtains a defense utility approximately the same as the equilibrium defense strategy (Eq) (Figures 3(d)(e)(f)). For the attacker's strategies, the sampled-activation attack strategy does not always obtain a high utility for the attacker. In fact, this attack strategy obtains the lowest attack utility in the case of low observation noise (Figure 3(b)). The solution quality of the value-propagation attack strategy, on the other hand, varies depending on the values of both the logistic parameter η^a and the percentage of candidates chosen to attack p^a. For example, when there is no observation noise (Figure 3(c)), aVP-0.5-3.0 and aVP-0.3-3.0 with $\eta^a = 3.0$ obtain a considerably higher attack utility compared to aVP-0.5-1.0 and aVP-0.3-1.0 with $\eta^a = 1.0$. On the other hand, in the case of low observation noise (Figure 3(b)), aVP-0.5-3.0 and aVP-0.5-1.0 with $per^a = 0.5$ obtain the highest attack utility compared with other attack strategies.

7 SUMMARY

In this work, we study the problem of deploying security countermeasures on Bayesian attack graphs to protect network data from cyber-attacks. We propose a new simultaneous multi-stage attack-graph security game model. Our game model encapsulates security environments with significant dynamics and uncertainty as a stochastic process over multiple time steps. We employ EGTA to analyze this complex security game, as obtaining an analytical



(a) Attacker, high noise (b) Attacker, low noise

(c) Attacker, no noise (d) Defender, high noise

(e) Defender, low noise (f) Defender, no noise

Figure 3: Strategy evaluation, random DAGs, 0% ∧-nodes

solution would be impractical. We propose different parameterized heuristic strategies for both players which leverage the topological structure of attack graphs and employ sampling to overcome the computational complexity. Our EGTA results on various game settings show that our defense heuristics not only outperform several baselines, but are also robust to defender uncertainty about graph states and the attacker's strategy.

Acknowledgment

This work was supported in part by MURI grant W911NF-13-1-0421 from the US Army Research Office.

A PROPAGATED DEFENSE REWARD

The idea of computing propagated defense rewards is similar to computing propagated attack values. In lines (3–12) of Algorithm 5, we compute the propagated defense reward $\hat{r}(u \mid \mathrm{S}_t)$ for all nodes $u \in \mathbf{V}$. In particular, $r^w(v \to u, t'+1)$ is the defense reward the postcondition v propagates to the precondition u with respect to the inactive goal node w and time step $t'+1$.

Algorithm 5: Compute Propagate Defense Reward

1 Input: time step, $t+1$, graph state, S_t, inverse topological order of $\mathbf{G}$, $itopo(\mathbf{G})$;
2 Initialize defense reward $r^w(v,t') = +\infty$ and $r^w(w,t') = r^d(w)$ for all nodes $v \in \mathbf{V} \setminus \{w\}$ and inactive goals $w \in \mathbf{V}^g \setminus \mathrm{S}_t$, for all time steps $t' \in \{t+1,\ldots,T\}$;
3 **for** $u \in itopo(\mathbf{G})$ **do**
4 **for** $v \in \pi^+(u) \setminus \mathrm{S}_t$ **do**
5 **for** $w \in \mathbf{V}^g \setminus (\mathrm{S}_t \cup \{u\}), t' \in \{t+1 \ldots T-1\}$ **do**
6 **if** $v \in \mathbf{V}^\wedge$ **then**
7 $r^w(v \to u, t'+1) = p(v) r^w(v,t')$;
8 **else**
9 $r^w(v \to u, t'+1) = p(u,v) r^w(v,t')$;
10 **if** $r^w(u,t'+1) > \gamma r^w(v \to u, t'+1)$ **then**
11 Update $r^w(u,t'+1) = \gamma r^w(v \to u, t'+1)$;
12 Return $\hat{r}(u \mid \mathrm{S}_t) = \min_{w \in \mathbf{V}^g \setminus \mathrm{S}_t} \min_{t' \in \{t+1,\ldots,T\}} r^w(u,t') (\neq +\infty), \forall u$;

REFERENCES

[1] S. Bistarelli, M. DallfiAglio, and P. Peretti. Strategic games on defense trees. In *International Workshop on Formal Aspects in Security and Trust*, pages 1–15. Springer, 2006.

[2] W. R. Cheswick, S. M. Bellovin, and A. D. Rubin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley, 2003.

[3] M. Dacier and Y. Deswarte. Privilege graph: An extension to the typed access matrix model. In *European Symposium on Research in Computer Security*, pages 319–334, 1994.

[4] M. Dacier, Y. Deswarte, and M. Kaâniche. Models and tools for quantitative assessment of operational security. In S. K. Katsikas and D. Gritzalis, editors, *Information Systems Security*, pages 179–186. Springer, 1996.

[5] S. Du, X. Li, J. Du, and H. Zhu. An attack-and-defence game for security assessment in vehicular ad hoc networks. *Peer-to-Peer Networking and Applications*, 7(3):215–228, 2014.

[6] K. Durkota, V. Lisỳ, B. Bošanskỳ, and C. Kiekintveld. Approximate solutions for attack graph games with imperfect information. In *6th International Conference on Decision and Game Theory for Security*, pages 228–249, 2015.

[7] K. Durkota, V. Lisỳ, B. Bošanskỳ, and C. Kiekintveld. Optimal network security hardening using attack graph games. In *24th International Joint Conference on Artificial Intelligence*, pages 526–532, 2015.

[8] D. Evans, A. Nguyen-Tuong, and J. Knight. Effectiveness of moving target defenses. In Jajodia et al. [10].

[9] M. Frigault, L. Wang, A. Singhal, and S. Jajodia. Measuring network security using dynamic Bayesian network. In *4th ACM Workshop on Quality of Protection*, pages 23–30, 2008.

[10] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, editors. *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*. Springer, 2011.

[11] B. Kordy, S. Mauw, M. Melissen, and P. Schweitzer. Attack–defense trees and two-player binary zero-sum extensive form games are equivalent. In *1st International Conference on Decision and Game Theory for Security*, pages 245–256. Springer, 2010.

[12] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer. DAG-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer Science Review*, 13:1–38, 2014.

[13] Y. Liu and H. Man. Network vulnerability assessment using Bayesian networks. In *Defense and Security*, pages 61–71. International Society for Optics and Photonics, 2005.

[14] R. D. McKelvey, A. M. McLennan, and T. L. Turocy. Gambit: Software tools for game theory. Technical report, Version 0.2006.01.20, 2006.

[15] E. Miehling, M. Rasouli, and D. Teneketzis. Optimal defense policies for partially observable spreading processes on Bayesian attack graphs. In *Second ACM Workshop on Moving Target Defense*, pages 67–76, 2015.

[16] A. K. Nandi, H. R. Medal, and S. Vadlamani. Interdicting attack graphs to protect organizations from cyber attacks: A bi-level defender–attacker model. *Computers & Operations Research*, 75:118–131, 2016.

[17] C. Phillips and L. P. Swiler. A graph-based system for network-vulnerability analysis. In *Workshop on New Security Paradigms*, pages 71–79. ACM, 1998.

[18] N. Poolsappasit, R. Dewri, and I. Ray. Dynamic security risk management using Bayesian attack graphs. *IEEE Transactions on Dependable and Secure Computing*, 9(1):61–74, 2012.

[19] United States Government Accountability Office. Information security, 2016. Report to Congressional Requesters.

[20] R. Van Der Merwe, A. Doucet, N. De Freitas, and E. A. Wan. The unscented particle filter. In *Advances in Neural Information Processing Systems*, pages 584–590, 2001.

[21] H. J. Wang, C. Guo, D. R. Simon, and A. Zugenmaier. Shield: Vulnerability-driven network filters for preventing known vulnerability exploits. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 193–204, 2004.

[22] M. P. Wellman. Putting the agent in agent-based modeling. *Autonomous Agents and Multi-Agent Systems*, 30:1175–1189, 2016.

[23] Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. In *6th International Conference on Mobile Computing and Networking*, pages 275–283, 2000.

[24] Q. Zhu and T. Başar. Game-theoretic approach to feedback-driven multi-stage moving target defense. In *4th International Conference on Decision and Game Theory for Security*, pages 246–263. Springer, 2013.

[25] S. A. Zonouz, H. Khurana, W. H. Sanders, and T. M. Yardley. RRE: A game-theoretic intrusion response and recovery engine. *IEEE Transactions on Parallel and Distributed Systems*, 25(2):395–406, 2014.