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

Thanh H. Nguyen[1], Mason Wright[2], Michael P. Wellman[2], Satinder Singh[2]
[1]University of Oregon, Eugene
thanhhng@cs.uoregon.edu
[2]University of Michigan, Ann Arbor
{masondw,wellman,baveja}@umich.edu

November 12, 2018

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. We propose parameterized heuristic strategies for the attacker and defender and provide detailed analysis of their time complexity. Our heuristics exploit the topological structure of attack graphs and employ sampling methods to overcome the computational complexity in predicting opponent actions. Due to the complexity of the game, we employ a simulation-based approach and perform empirical game analysis over an enumerated set of heuristic strategies. Finally, we conduct experiments in various game settings to evaluate the performance of our heuristics in defending networks, in a manner that is robust to uncertainty about the security environment.[1]

1 Introduction

Attack graphs are graphical models used in cybersecurity research to decompose complex security scenarios into a hierarchy of simple and quantifiable actions [Dacier et al., 1996, Dacier and Deswarte, 1994, Phillips and Swiler, 1998]. Many research efforts have employed attack graphs or related graphical security models to analyze complex security scenarios [Albanese and Jajodia, 2018, Kordy et al., 2014]. In particular, attack-graph models have been used to evaluate network hardening strategies, where a network administrator (the defender) deploys security countermeasures to protect network data from cybercriminals (the attacker) [Bistarelli et al., 2006, Du et al., 2014, Durkota et al., 2015a,b, Kordy et al., 2010, Miehling et al., 2015, Nandi et al., 2016, Zonouz et al., 2014].

Attack graphs are particularly suitable for modeling scenarios in moving target defense (MTD) [Jajodia et al., 2011], where the defender employs proactive tactics to dynamically change network configurations, limiting the exposure of vulnerabilities. MTD techniques are most useful in thwarting progressive attacks [Evans et al., 2011, Prakash and Wellman, 2015, Zhu and Başar, 2013], where system reconfiguration by the defender prevents the attacker from exploiting knowledge accumulated over time. Attack graphs naturally represent the progress of an attack, and the defense actions in our model can incorporate MTD methods.

We build on prior works that represent security problems with attack graphs, and we take a game-theoretic approach to reasoning about the strategic interaction between the defender and the attacker. Building on an existing Bayesian attack graph formalism [Miehling et al., 2015, Poolsappasit et al., 2012], we model

[1]This manuscript is an extension of a paper in the proceedings of the 2017 Workshop on Moving Target Defense [Nguyen et al., 2017].

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 can 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, 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 considered in its strategic reasoning.

Based on our game model, we propose various parameterized strategies for both players. Our attack strategies assess the value of each possible attack action at a given time step by examining the future attack paths they enable. These paths are sequences of nodes which could feasibly be attacked in subsequent 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, our attack heuristics incorporate two simplifying approximations. 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. 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. We present a detailed analysis of the polynomial time complexity of our attack heuristics.

Likewise, our heuristic defense strategies employ simplifying assumptions. At each time step, the defense strategies: (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 [van der Merwe et al., 2001] 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. We show that the running time of our defense heuristics is also polynomial.

Finally, we employ a simulation-based methodology called *empirical game-theoretic analysis* (EGTA) [Wellman, 2016], to construct and analyze game models over the heuristic strategies. We present a 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 states and about the attacker's strategy.

2 Related work

Attack graphs are commonly used to provide a convenient representation for analysis of network vulnerabilities [Phillips and Swiler, 1998, Dacier et al., 1996, Dacier and Deswarte, 1994]. In an attack graph, nodes represent attack conditions of a system, and edges represent relationships among these conditions: specifically, how the achievement of specific conditions through an attacker's actions can enable other conditions. Various graph-based security models related to attack graphs have been proposed to represent and analyze complex security scenarios [Kordy et al., 2014]. For example, Wang et al. [2006] introduced such a model for correlating, hypothesizing, and predicting intrusion alerts. *Temporal attack graphs*, introduced by Albanese and Jajodia [2018], extend the attack-graph model of Wang et al. [2006] with additional temporal constraints on the unfolding of attacks. *Augmented attack trees* were introduced by Ray and Poolsapassit [2005] to probabilistically measure the progression of an attacker's actions toward compromising the system. This concept of augmented attack trees was later used to perform a forensic analysis of log files [Poolsapassit and Ray, 2007]. Vidalis et al. [2003] presented *vulnerability trees* to capture the interdependency between different vulnerabilities of a system. *Bayesian attack graphs* [Liu and Man, 2005] combine attack graphs with quantified uncertainty on attack states and relations. Revised versions of Bayesian attack graphs incorporate other security aspects such as dynamic behavior and mitigation strategies [Frigault et al., 2008, Poolsappasit et al., 2012]. Our work is based on the specific formulation of Bayesian attack graphs by Poolsappasit et al. [2012]. The basic ideas of our game model and heuristic strategies would also apply to variant forms of graph-based security models with with reasonable modifications. In particular, our work might be extended to handle zero-day attacks captured by zero-day attack graphs [Kotenko and Chechulin, 2013, Wang et al., 2010, 2014, Ingols et al., 2009] by incorporating *payoff uncertainty* as a result of unknown locations and impacts of zero-day vulnerabilities. In the scope of this work, we consider known payoffs of players.

Though many attack-graph models attempt to analyze different progressive attack scenarios without considering countermeasures of a defender, there is an important line of work on graph-based models covering both attacks and defenses [Kordy et al., 2014]. For example, Wu et al. introduced *intrusion DAGs* to represent the underlying structure of attacker goals in an adaptive, intrusion-tolerant system. Each node of an intrusion DAG is associated with an alert from the intrusion-detection framework, allowing the system to automatically trigger a response. *Attack-Response Trees (ARTs)*, introduced by Zonouz et al. [2014], extend attack trees to incorporate possible response actions against attacks.

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. [2015a,b] study the problem of hardening the security of a network by deploying honeypots to the network to deceive the attacker. They model the problem as a Stackelberg security game in which the attacker's plans are compactly represented using attack graphs. Zonouz et al. [2014] 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. 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 [Bistarelli et al., 2006, Du et al., 2014, Kordy et al., 2010].

Game theory has been applied for solving various cybersecurity problems [Alpcan and Basar, 2004, Zhu and Rass, 2018, Lye and Wing, 2005, Nguyen et al., 2009b, Alpcan and Basar, 2006, Nguyen et al., 2009a, Patcha and Park, 2004, You and Shiyong, 2003]. Previous work has modeled these security problems as a dynamic (complete/incomplete/imperfect) game and analyzed equilibrium solutions of those games. Our game model (to represent security problems with attack graphs) belongs to the class of partially observable stochastic games. Since the game is too complex for analytic solution, we focus on developing heuristic strategies for players and employing the simulation-based methodology EGTA to evaluate these strategies.

Similar to our work, the non-game-theoretic solution proposed by Miehling et al. [2015] is also built on the attack-graph formalism of Poolsappasit et al. [2012]. In their work, the attacker's behavior is modeled by a probabilistic spreading process, which is known by the defender. In our model, on the other hand, 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 [Poolsappasit et al., 2012]). *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-related attributes of the network systems including (i) system vulnerabilities; (ii) insecure system properties such as corrupted files or memory access permissions; (iii) insecure network properties such as unsafe network conditions or unsafe firewall properties; and (iv) access privilege conditions such as user account or root account.*

- *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. For example, suppose a node (representing the root access privilege on a machine) is active, it means the attacker has the root access privilege on that machine.

- $\mathbf{E}$ *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. *For example, suppose a node u represents the sshd BOF vulnerability on machine A and node v represents the root access privilege on A. Then the exploit (u, v) indicates that the attacker can exploit the sshd BOF vulnerability on A to obtain the root access privilege on A. 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 (assuming v is not defended, u is active, and no other exploit (u', v) is attacked). $\wedge$-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 (assuming v is not defended and all parent nodes of v are active).*

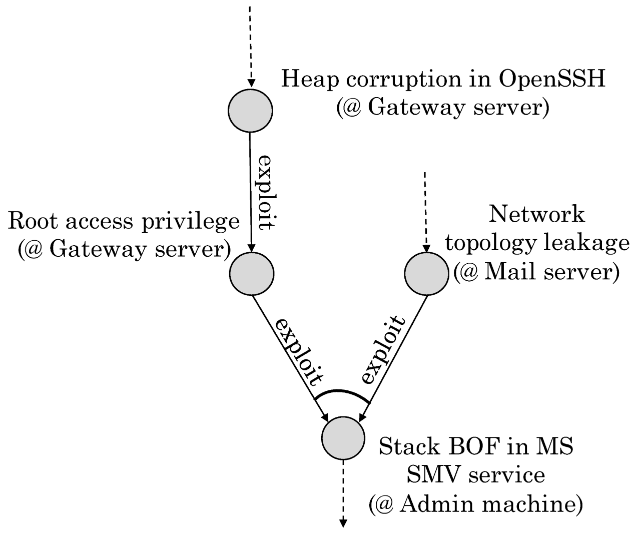


Figure 1: Portion of Bayesian attack graph for testbed network of Poolsappasit et al. [2012]. The node with exploits connected by an arc is $\wedge$-type; the rest are $\vee$-type. Dotted edges indicate excluded portions of the graph. In this graph portion, there are four nodes representing system vulnerabilities and access privileges. The attacker can exploit the heap corruption in OpenSSH at the gateway server to obtain root access privileges. Then the attacker can exploit root access privileges at the gateway server and the network topology leakage at the mail server to cause the stack BOF in the MS SMV service of the admin machine.

An example of a Bayesian attack graph based on this definition is shown in Figure 1. Our multi-stage Bayesian attack graph security game model is defined as follows.

Definition 3.2 (Attack Graph Game)**.** *A Bayesian 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* $p(1_v \mid s_t(v) = 1) \in (0, 1]$*, 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 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. An example attack graph is shown in Figure 2.

3.2 Network example

We first briefly present the test-bed network introduced by Poolsappasit et al. [2012]. We then describe a portion of the corresponding Bayesian attack graph with security controls of the network. Our security game model and proposed heuristic strategies for both the defender and attacker are built based on their model.

Overall, the test-bed network consists of eight hosts located within two different subnets: DMZ zone and Trusted zone. The DMZ zone includes a mail server, a DNS server, and a web server. The Trusted zone has two local desktops, an administrative server, a gateway server, and an SQR server. There is an installed tri-homed DMZ firewall with a set of policies to separate servers in the DMZ network from the local network. The attacker has to pass the DMZ firewall to attack the network. The web server in the DMZ zone can send SQL queries to the SQL server in the Trusted zone on a designated channel. In the Trusted zone, there is a NAT firewall such that local machines (located behind NAT) have to communicate with external parties through the gateway server. Finally, the gateway server monitors remote connections through SSHD.

There are several vulnerabilities associated with each machine which can be exploited by the attacker in the test-bed network. For example, the gateway server has to face with *heap corruption in OpenSSH* and

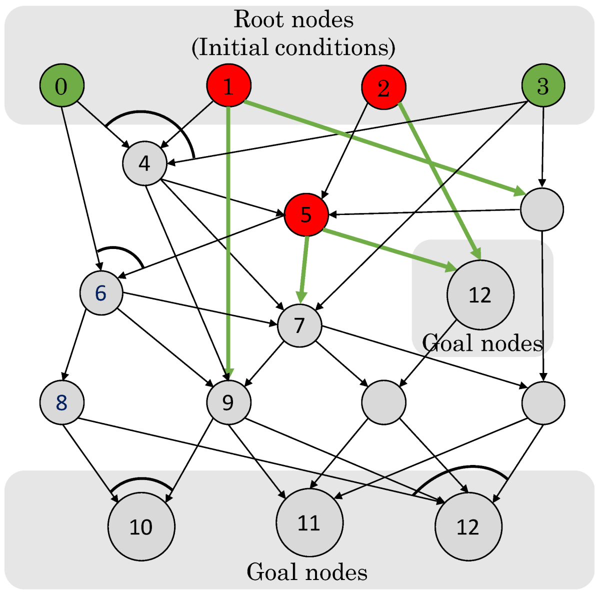


Figure 2: Bayesian attack graph example. Nodes with incoming edges connected by black curves are $\wedge$-type. The others are $\vee$-type. For instance, activating $\wedge$-type node 4 requires all exploits from nodes 0, 1, and 3 to 4 to be feasible and taken by the attacker. At the current time step, red nodes 1, 2, and 5 are active. Other gray and green nodes are inactive. Thus, the attacker's action can be any subset of green nodes and green exploits. For example, the attacker can directly activate the root nodes 0 and 3. The attacker can also activate node 9 by taking the feasible exploit $(1, 9)$. Conversely, the defender can choose any subset of nodes to protect. Suppose the attacker decides to activate node 0 and node 9 (via exploit $(1, 9)$) while the defender decides to protect nodes 0 and 7. Then node 0 remains inactive. Node 9 becomes active with an activation probability associated with exploit $(1, 9)$.

improper cookie handler in OpenSSH. In addition, SQL and DNS servers have to deal with the *SQR injection* and *DNS cache poisoning* vulnerabilities respectively. The defender can deploy security controls such as *limit access to DNS server* to tackle the vulnerability *DNS cache poisoning* [Poolsappasit et al., 2012].

A portion of the Bayesian attack graph of the testbed network is shown in Figure 3. A complete Bayesian attack graph can be found in Poolsappasit et al. [2012]. Each grey rectangle represents a security attribute of the network. For example, the attacker can exploit the *stack BOF* at local desktops to obtain the *root access privilege* at those desktops. Then by exploiting the *root access privilege*, the attacker can achieve the *error message leakage* at the mail server and the *DNS cache poisoning* at the DNS server. Finally, the attacker can obtain the *identity theft* and *information leakage* by exploiting the *error message leakage* at the mail server, etc. To prevent such attack progression, the defender can deploy the *MS workaround* to deal with the vulnerability *stack BOF*. The defender can also deploy *POP3* and *limit access to DNS server* to address the vulnerabilities *error mesage leakage* and *DNS cache poisoning* respectively. Finally, *encryption* and *digital signature* could be used to resolve the *identity theft* issues at the DNS and mail server.

3.3 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 $\mathbf{S}_t$.

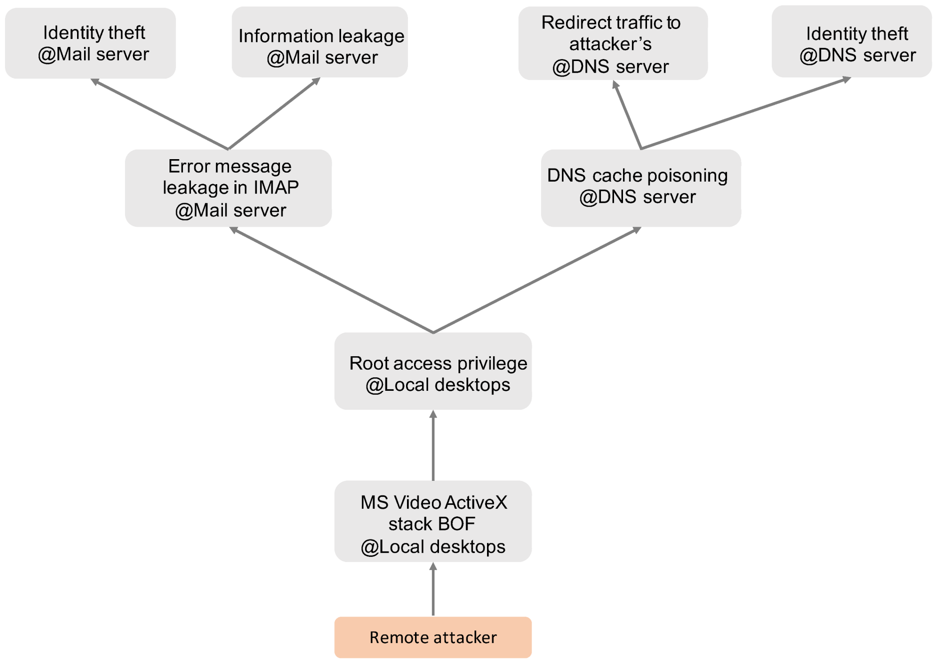

Identity theft @Mail server
Information leakage @Mail server
Redirect traffic to attacker's @DNS server
Identity theft @DNS server
Error message leakage in IMAP @Mail server
DNS cache poisoning @DNS server
Root access privilege @Local desktops
MS Video ActiveX stack BOF @Local desktops
Remote attacker

Figure 3: A portion of the Bayesian attack graph of the testbed network in Poolsappasit et al. [2012].

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

```
1 Initialize $s_{t+1}(v) \leftarrow s_t(v)$;
2 if $v \in \mathbf{D}_{t+1}$ then
3     $s_{t+1}(v) \leftarrow 0$;                         // defender overrules attacker
4 else
5     if $v \in \mathbf{A}_{t+1} \cap \mathbf{V}^{\wedge}$ and $\pi^{-}(v) \subseteq \mathbf{S}_t$ then
6         with probability $p(v)$, $s_{t+1}(v) \leftarrow 1$
7     else
8         for $(u, v) \in \mathbf{A}_{t+1} \cap \mathbf{E}^{\vee}$ and $u \in \mathbf{S}_t$ do
9             with probability $p(u, v)$, $s_{t+1}(v) \leftarrow 1$
```

- 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.

3.4 Payoff function

We denote by $\mathbf{\Omega}_T = \{(\mathbf{A}_0, \mathbf{D}_0, \mathbf{S}_0), \ldots, (\mathbf{A}_T, \mathbf{D}_T, \mathbf{S}_T)\}$ 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 $\mathbf{\Omega}_T$, which comprise goal rewards and action costs, are computed as follows:

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

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

Both players aim to maximize expected utility with respect to the distribution of $\mathbf{\Omega}_T$. 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. Our heuristic strategies for each player are categorized based on: (i) the assumptions regarding their opponent's strategies; and (ii) heuristic methods used to generate actions for the players to take at each time step. In particular, our proposed heuristic strategies can be explained in the following hierarchical view.

4 Level-0 Heuristic Strategies

We define a *level-0 strategy* as one that does not explicitly invoke an assumption about its opponent's strategy. Higher-level strategies do invoke such assumptions, generally that the others play one level below, in the spirit of cognitive hierarchy models [Camerer et al., 2004]. In our context, level-0 heuristic strategies directly operate on the topological structure of the Bayesian attack graph to decide on actions to take at each time step.

4.1 Level-0 defense strategies

We introduce four level-0 defense strategies. Each targets a specific group of nodes to disable at each time step $t+1$. These groups of nodes are chosen solely based on the topological structure of the graph.

4.1.1 Level-0 uniform defense strategy

The defender chooses nodes in the graph to disable uniformly at random. The number of nodes chosen is a certain fraction of the total number of nodes in the graph.

4.1.2 Level-0 min-cut uniform defense strategy

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

4.1.3 Level-0 root-node uniform defense strategy

The defender chooses root nodes to disable uniformly at random. The number of nodes chosen is a certain fraction of the total number of root nodes in the graph.

4.1.4 Level-0 goal-node defense 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 goal node $v \in \mathbf{V}^g$ is based on the conditional logistic function:

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

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. This parameter governs how strictly the choice follows assessed defense values. In particular, if $\eta^d = 0$, the goal-node defense strategy chooses to disable each goal node uniformly at random. On the other hand, if $\eta^d = +\infty$, this strategy only disables nodes with highest $\gamma^t(-r^d(v) + c^d(v))$. 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.

4.2 Level-0 attack strategies

4.2.1 Attack candidate set

At time step $t+1$, based on the graph state $\mathbf{S}_t$, 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:

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

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. Each $\wedge$-node or $\vee$-exploit in $\mathbf{\Psi}^a(\mathbf{S}_t)$ is considered as a candidate attack at $t+1$. An attack action at $t+1$ can be any subset of this candidate set. For example, in Figure 2, the current graph state is $\mathbf{S}_t = \{1, 2, 5\}$. The attack candidate set thus consists of (i) all green edges; and (ii) all green nodes. In particular, the attacker can perform exploit $(1, 9)$ to activate the currently inactive node 9. The attacker can also attempt to activate the root nodes 0 and 3.

To find an optimal attack action to take at $t+1$, we need to determine the attack value of each possible attack action at time step $t+1$, which represents the attack action's impact on activating the goal nodes by the final time step T. However, exactly computing the attack value of each attack action requires taking into account all possible future outcomes regarding this attack action, which is computationally expensive. In the following, we propose a series of heuristic attack strategies, of increasing complexity.

4.2.2 Level-0 uniform attack strategy

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

4.2.3 Level-0 value-propagation attack strategy

Attack value propagation. The value-propagation strategy chooses attack actions based on a quantitative assessment of each attack in the candidate set $\mathbf{\Psi}^a(\mathbf{S}_t)$. The main idea of this strategy is to approximate the attack value of each individual inactive node locally based on attack values of its inactive postconditions. Attack values of the inactive goal nodes, in particular, correspond to the attacker's rewards at these nodes. So essentially, the attacker rewards $r^a(w) > 0$ at inactive goal nodes $w \in \mathbf{V}^g \setminus \mathbf{S}_t$ are propagated 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 propagation paths reaching that node. This propagation process is illustrated in Algorithm 2, which approximates attack values of every inactive node in polynomial time.

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 $r^w(v,t') \leftarrow 0$ and $r^w(w,t+1) \leftarrow r^a(w)$ for inactive goal nodes $w \in \mathbf{V}^g \setminus \mathbf{S}_t$, all
   inactive nodes $v \in \mathbf{V} \setminus (\{w\} \cup \mathbf{S}_t)$, 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 $w \in \mathbf{V}^g \setminus (\mathbf{S}_t \cup \{u\}), t' \leftarrow t+1, \ldots, T-1$ do
6              if $v \in \mathbf{V}^\wedge$ then
7                  $r^w(v \rightarrow u, t'+1) \leftarrow \frac{c^a(v)+p(v)r^w(v,t')}{|\pi^-(v)\setminus\mathbf{S}_t|^\alpha}$;
8              else
9                  $r^w(v \rightarrow u, t'+1) \leftarrow c^a(u,v) + p(u,v)r^w(v,t')$;
10             if $r^w(u,t'+1) < \gamma r^w(v \rightarrow u, t'+1)$ then
11                 Update $r^w(u,t'+1) \leftarrow \gamma r^w(v \rightarrow u, t'+1)$;
12 Return $\hat{r}(u) \leftarrow \max_{w\in\mathbf{V}^g\setminus\mathbf{S}_t} \max_{t'\in\{t+1,\ldots,T\}} r^w(u,t')$, $\forall u \in \mathbf{V} \setminus \mathbf{S}_t$;
```

Algorithm 2 leverages the directed acyclic topological structure of the Bayesian attack graph to perform the goal-value propagation faster. 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 the 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, $r^w(v \rightarrow u, t'+1)$. If node v is of $\wedge$-type, the propagated 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 propagated 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). An example of Algorithm 2 is illustrated in Figure 4.

Proposition 1. *The time complexity of Algorithm 2 is* $O((|\mathbf{V}| + |\mathbf{E}|) \times |\mathbf{V}^g| \times |\mathbf{T}|)$.

Proof. In Algorithm 2, line 2 initializes attack values of inactive nodes of the attack graph with respect to each inactive goal node and time step. The time complexity of this step is $O(|\mathbf{T}| \times |\mathbf{V}| \times |\mathbf{V}^g|)$. Algorithm 2 then iteratively examines each inactive node once following the inverse direction of the topological order. The attack value of each node with respect to each inactive goal node and time step is estimated locally based on its neighboring nodes. In other words, Algorithm 2 iterates over each edge of the graph once to compute this attack value. The time complexity of this step is thus $O(|\mathbf{E}| \times |\mathbf{V}^g| \times |\mathbf{T}|)$. Finally, line 12 computes the maximum propagated attack value for each node, which takes $O(|\mathbf{T}| \times |\mathbf{V}| \times |\mathbf{V}^g|)$ time. Therefore, the total time complexity of Algorithm 2 is $O((|\mathbf{V}| + |\mathbf{E}|) \times |\mathbf{V}^g| \times |\mathbf{T}|)$. □

Probabilitic selection of attack action. Based on attack values of inactive nodes, we approximate the value of each candidate attack in $\mathbf{\Psi}^a(\mathbf{S}_t)$ taking into account the cost of this attack and the corresponding

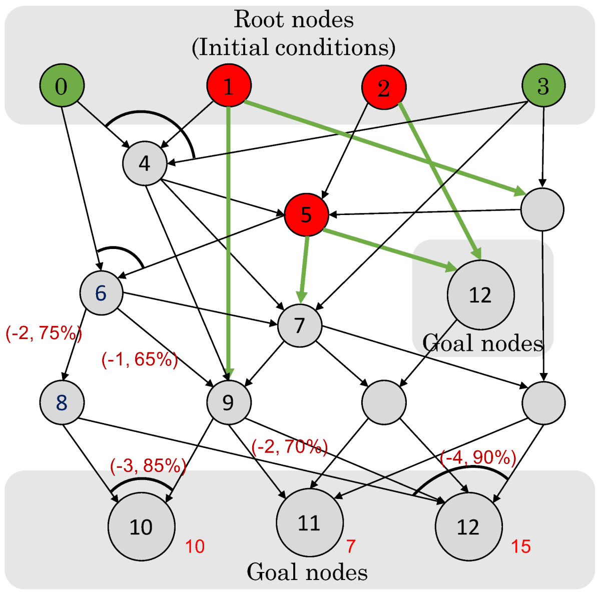


Figure 4: Level-0 value-propagation attack strategy. In this example, the discount factor is $\gamma = 0.9$. The attacker rewards at goal nodes are $r^a(10) = 10, r^a(11) = 7, r^a(12) = 15$. If the attacker performs exploits $(8, 10)$ and $(9, 10)$ to activate the $\wedge$-node 10, he has to pay a cost of $c^a(10) = -3$ and the corresponding activation probability is $p(10) = 0.85$. In addition, if the attacker performs the exploit $(9, 11)$ to activate the goal node 11, his cost is $c^a(9, 11) = -2$ and the activation probability is $p(9, 11) = 0.7$. Algorithm 2 works as follows. Suppose that the propagation parameter $\alpha = 1$, then Algorithm 2 estimates the propagated attack value from postconditions 10 and 12 to precondition 8 as $r^{10}(10 \to 8, t+2) = \frac{-3+0.85\times 10}{2}$ and $r^{12}(12 \to 8, t+2) = \frac{-4+0.90\times 15}{4}$. Similarly, Algorithm 2 estimates the propagated attack value from postconditions 10, 11 and 12 to precondition 9 as $r^{10}(10 \to 9, t+2) = \frac{-3+0.85\times 10}{2}$, $r^{11}(11 \to 9, t+2) = -2 + 0.7 \times 7$, and $r^{12}(12 \to 9, t+2) = \frac{-4+0.90\times 15}{4}$. The attack values of other nodes of which postconditions are the goal nodes are estimated similarly. The attack value of node 6 will be computed based on the attack value of its postconditions 7, 8, and 9, and so on. This process will continue until all inactive nodes are examined.

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 attack strategy selects attacks to execute probabilistically, based on the assessed attack value. It first determines the number of attacks to execute. The strategy then selects that number of attacks using a conditional logistic function. The probability each exploit $e \in \mathbf{\Psi}^a(\mathbf{S}_t)$ is selected is computed 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. The model parameter $\eta^a \in [0, +\infty)$ governs how strictly the choice follows assessed attack values.

4.2.4 Level-0 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 propagating backward from goal nodes, this strategy constructs estimates by forward sampling from the current candidates $\mathbf{\Psi}^a(\mathbf{S}_t)$.

Random activation process. The sampled-activation attack strategy aims to sample paths of activation from the current graph state $\mathbf{S}_t$ to activate each inactive goal node. For example, in Figure 4, given the current graph state is $\mathbf{S}_t = \{1, 2, 5\}$, a possible path or sequence of nodes in order to activate the inactive goal node 10 is as follows: (i) activate node 0; (ii) if 0 becomes active, perform exploits $(0, 6)$ and $(5, 6)$ to activate node 6; (iii) if 6 becomes active, perform exploits $(6, 8)$ and $(6, 9)$ to activate nodes 8 and 9; and (iv) if nodes 8 and 9 become active, perform exploits $(8, 10)$ and $(9, 10)$ to activate goal node 10. In fact, there are many possible paths which can be selected to activate each inactive goal node of the graph. Therefore, the sampled-activation attack strategy selects each path probabilistically based on the probability that each inactive goal node will become active if the attacker follows that path to activate the goal node.

Algorithm 3: Random Activation

```
1  Input: $t+1$, $\mathbf{S}_t$, and topological order of $\mathbf{G}$, $topo(\mathbf{G})$;
2  Initialize $p^{act}(v) \leftarrow 1.0$, $t^{act}(v) \leftarrow t$, and $pre(v) \leftarrow \emptyset$, for all active nodes $v \in \mathbf{S}_t$;
3  Initialize $p^{act}(v) \leftarrow p(v)$, $t^{act} \leftarrow t+1$, and $pre(v) \leftarrow \emptyset$ 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
           $p^{ra}(u, v) \propto p^{act}(u)p(u, v), \forall u \in \pi^-(v)$;
7          Update $p^{act}(v) \leftarrow p^{act}(u)p(u, v)$;
8          Update $t^{act}(v) \leftarrow t^{act}(u) + 1$;
9          Update $pre(v) \leftarrow \{u\}$;
10     else
11         Update $p^{act}(v)$ with respect to all preconditions $\pi^-(v)$;
12         Update $t^{act}(v) \leftarrow \max_{u \in \pi^-(v)} t^{act}(u) + 1$;
13         Update $pre(v) \leftarrow \pi^-(v)$;
14 Return $\{(p^{act}(v), t^{act}(v), pre(v))\}$;
```

In particular, for each node v, the sampled-activation attack strategy keeps 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. We select only one precondition to activate each inactive $\vee$-node to simplify the computation of probability a node becomes active in the random activation. 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 becomes active if the attacker follows the sampled action sequence to activate v. These values are computed under the assumptions that the defender takes no action and the attacker attempts to activate each node on activation paths once.

The random activation process is illustrated in Algorithm 3. In this process, we use the topological order of the attack graph to perform random activation. Following this order ensures that all preconditions are visited before any corresponding postconditions and thus we only need to examine each node once. 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, $p^{act}(u)p(u, v)$ is the probability v becomes active if the attacker chooses the exploit (u, v) to activate v in the random activation. Accordingly, the higher $p^{act}(u)p(u, v)$ is, the higher the chance that u is the selected precondition for v. We 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). Furthermore, v can become active only when all of its preconditions are active. Thus, 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 all the preconditions u of v. 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) which can be defined as follows:

$$\mathit{seq}(v) \equiv \{v\} \cup \mathit{pre}(v) \cup \mathit{pre}(\mathit{pre}(v)) \cdots$$

For example, in Figure 4, given the current graph state is $\mathbf{S}_t = \{1, 2, 5\}$, we suppose that the sequence of nodes chosen to activate the inactive goal node 10 is: (i) activate node 0; (ii) if 0 becomes active, perform exploits $(0, 6)$ and $(5, 6)$ to activate nodes 6; (iii) if 6 becomes active, perform exploits $(6, 8)$ and $(6, 9)$ to activate nodes 8 and 9; and (iv) if nodes 8 and 9 become active, perform exploits $(8, 10)$ and $(9, 10)$ to activate goal node 10. Thus, $\mathit{seq}(10) = \{10\} \cup \{8, 9\} \cup \{6\} \cup \{0, 5\}$. Essentially, following the random activation process, v can be activated only when all nodes in $\mathit{seq}(v) \setminus \{v\}$ are active. Therefore, the activation probability, $p^{act}(v)$, is computed as follows, which comprises the activation probabilities of all edges and nodes involved in activating v:

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

where $\mathit{seq}(v) = \mathit{seq}^{\vee}(v) \cup \mathit{seq}^{\wedge}(v)$, $\mathit{seq}^{\vee}(v)$ consists of $\vee$-nodes only and $\mathit{seq}^{\wedge}(v)$ consists of $\wedge$-nodes.

Proposition 2. *The time complexity of Algorithm 3 is* $O(|\mathbf{V}| \times (|\mathbf{V}| + |\mathbf{E}|))$.

Proof. In the random activation process, for each visited node $v \in \mathbf{V}^{\vee}$, Algorithm 3 updates the activation probability $p^{act}(v)$ and activation time $t^{act}(v)$ based on the preconditions of v. The complexity of updating all nodes $v \in \mathbf{V}^{\vee}$ is thus $O(|\mathbf{V}^{\vee}| + |\mathbf{E}^{\vee}|)$. On the other hand, for each visited node $v \in \mathbf{V}^{\wedge}$, Algorithm 3 backtracks all nodes in the sequence of activating v, of which complexity is $O(|\mathbf{V}| + |\mathbf{E}|)$. Updating all nodes $v \in \mathbf{V}^{\wedge}$ is thus $O(|\mathbf{V}^{\wedge}| \times (|\mathbf{V}| + |\mathbf{E}|))$. Therefore, the time complexity of Algorithm 3 is $O(|\mathbf{V}| \times (|\mathbf{V}| + |\mathbf{E}|))$. $\square$

Expected utility of the attacker. At the end of a random activation, we obtain a sequence of nodes chosen to activate each inactive goal node. Thus, we 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 based on Proposition 3:

Proposition 3. *At time step $t + 1$, given the graph state $\mathbf{S}_t$, we suppose the attacker follows a random activation process to activate a subset of goal nodes $\hat{\mathbf{V}}^g \subseteq \mathbf{V}^g \setminus \mathbf{S}_t$. If the defender takes no further action, the attacker obtains an expected utility which is computed as follows:*

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

where $\mathit{seq}^{\wedge}(\hat{\mathbf{V}}^g) = \cup_{v \in \hat{\mathbf{V}}^g} \mathit{seq}^{\wedge}(v)$ and $\mathit{seq}^{\vee}(\hat{\mathbf{V}}^g) = \cup_{v \in \hat{\mathbf{V}}^g} \mathit{seq}^{\vee}(v)$ consist of all $\wedge$-nodes and $\vee$-nodes in the sequences chosen by the random activation process to activate inactive goal nodes in $\hat{\mathbf{V}}^g$.

Proof. In this equation, the first term accounts for the expected rewards of the goal nodes in the subset. In particular, for each goal node $v \in \hat{\mathbf{V}}^g$, the probability v becomes active at time step $t^{act}(v)$ if the attacker follows the random activation process is $p^{act}(v)$. Conversely, node v remains inactive. Therefore, the attacker receives an expected reward of $p^{act}(v) r^a(v) \gamma^{t^{act}(v)-1}$ regarding each goal node $v \in \hat{\mathbf{V}}^g$. Furthermore, second and third terms account for the costs of activating inactive nodes in the corresponding sampled-activation sequences. 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 $c^a(pre(v), v)$. $\square$

Greedy Attack. For each random activation, the sampled-activation attack strategy aims to find a subset of inactive goal nodes to activate 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. 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 the attack value $r(\hat{\mathbf{V}}^g \cup \{u\})$ 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 chosen $\hat{\mathbf{V}}^g$, we obtain a corresponding candidate subset:

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

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.

Finally, by running random activation multiple times, we obtain a set of candidate subsets, each associated with an estimated attack value. 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.

Proposition 4. *Suppose that the sampled-activation attack strategy runs random activation N^r times, the time complexity of this attack strategy is $O\left(N^r \times (|\mathbf{V}| + |\mathbf{E}|) \times (|\mathbf{V}^g|^2 + |\mathbf{V}|)\right)$.*

Proof. For each random activation, it takes $O(|\mathbf{V}| \times (|\mathbf{V}| + |\mathbf{E}|))$ time to sample activation paths for the attacker to activate each goal node (Proposition 2). Furthermore, at each iteration of the greedy attack heuristic, given current chosen goal subset $\hat{\mathbf{V}}^g$, the strategy examines all inactive goal nodes to find the next best goal node u. Each such examination of an inactive goal node u requires computing the expected utility of the attacker to follow the random activation to activate goal nodes in $\hat{\mathbf{V}}^g \cup \{u\}$, which takes $O(|\mathbf{V}| + |\mathbf{E}|)$ time (Proposition 3). Thus, the time complexity of each iteration of the greedy attack heuristic is $O((|\mathbf{V}| + |\mathbf{E}|) \times |\mathbf{V}^g|)$. As a result, the time complexity of the greedy attack heuristic is $O\left((|\mathbf{V}| + |\mathbf{E}|) \times |\mathbf{V}^g|^2\right)$. The sampled-activation attack strategy comprises of N^r pairs of executing random activation and greedy attack heuristic. Therefore, the complexity of this attack strategy is $O\left(N^r \times (|\mathbf{V}| + |\mathbf{E}|) \times (|\mathbf{V}^g|^2 + |\mathbf{V}|)\right)$. $\square$

While level-0 heuristic strategies of each player do not take into account their opponent's strategies, we introduce level-1 heuristic strategies, assuming the opponents play level-0 strategies. In the scope of this work, we focus on studying level-1 heuristic defense strategies.

5 Level-1 Defense Strategies

5.1 Defender strategic reasoning

Level-1 defense strategies assume the attacker plays level-0 attack strategies. At each time step, because the defender does not know the true graph state, it is important for it to reason about possible graph states before choosing defense actions. An overview of the defender belief update is shown in Figure 5.

As mentioned before, in our game, the defender knows the initial graph state, in which all nodes are inactive. As the game evolves, at the end of each time step t, the defender updates her belief on possible graph states, taking into account (i) her belief on graph states at the end of time step $t-1$; (ii) her action at time step t; (iii) her observations at time step t after the players' actions at t are taken; and (iv) the assumed attacker strategy. Finally, based on the defender's belief update on graph states at the end of time step t and the assumption of which heuristic strategy the attacker plays, the defender decides on which defense action to take at time step $t+1$. 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.

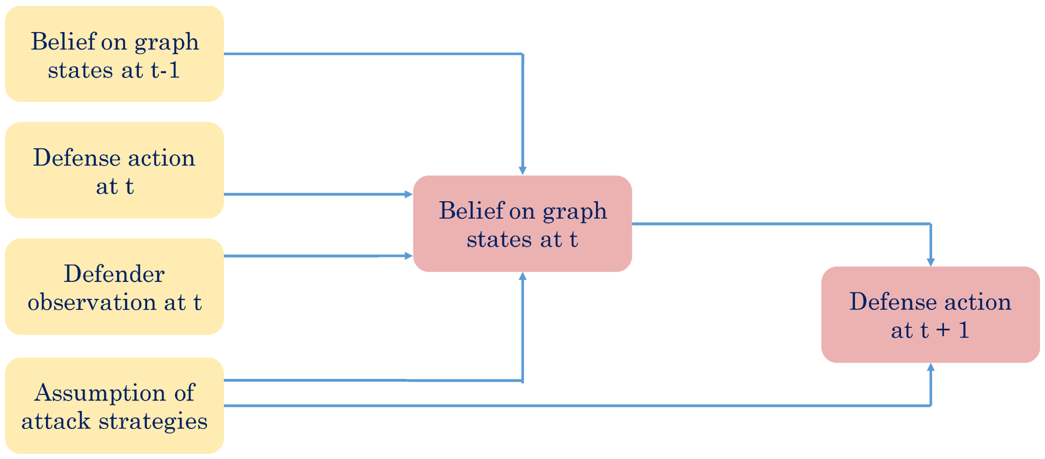

Belief on graph states at t-1
Defense action at t
Defender observation at t
Assumption of attack strategies
Belief on graph states at t
Defense action at t + 1

Figure 5: Overview of the defender belief update on graph states

5.2 Defender belief update

5.2.1 Exact inference

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 at time step $t-1$, $\mathbf{b}_{t-1}$, her action at t, $\mathbf{D}_t$, and her observation at t, $\mathbf{O}_t$, we can update the defender's belief $\mathbf{b}_t$ based on Bayes' rule as follows:

$$b_t(\mathbf{S}_t) = p(\mathbf{S}_t \mid \mathbf{b}_{t-1}, \mathbf{D}_t, \mathbf{O}_t) \propto p(\mathbf{S}_t, \mathbf{b}_{t-1}, \mathbf{D}_t, \mathbf{O}_t) \tag{1}$$
$$\propto p(\mathbf{O}_t \mid \mathbf{S}_t)\left[\sum_{\mathbf{S}_{t-1}\in\mathfrak{S}(\mathbf{b}_{t-1})} b_{t-1}(\mathbf{S}_{t-1}) \sum_{\mathbf{A}_t\in\mathfrak{A}(\mathbf{S}_{t-1})} p(\mathbf{S}_t \mid \mathbf{A}_t, \mathbf{D}_t, \mathbf{S}_{t-1}) p(\mathbf{A}_t \mid \mathbf{S}_{t-1})\right]$$

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 based on the observation probability of each node, 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, $\mathfrak{S}(\mathbf{b}_{t-1})$ is the belief state set associated with the defender's belief $\mathbf{b}_{t-1}$ at time step $t-1$: $\mathfrak{S}(\mathbf{b}_{t-1}) = \{\mathbf{S}_{t-1} \mid b_{t-1}(\mathbf{S}_{t-1}) > 0\}$. The probability of state transition $p(\mathbf{S}_t \mid \mathbf{A}_t, \mathbf{D}_t, \mathbf{S}_{t-1})$ is computed based on the state transition of every node:

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

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

5.2.2 Approximate inference

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 [van der Merwe et al., 2001], a Monte Carlo sampling method for performing state inference, given noisy observations

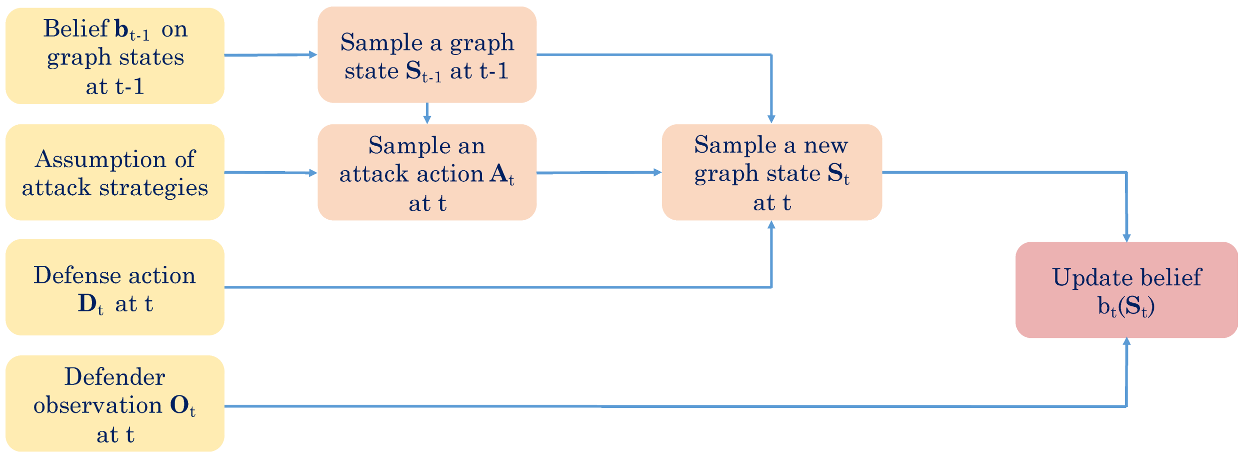

Belief $\mathbf{b}_{t-1}$ on graph states at t-1
Assumption of attack strategies
Defense action $\mathbf{D}_t$ at t
Defender observation $\mathbf{O}_t$ at t
Sample a graph state $\mathbf{S}_{t-1}$ at t-1
Sample an attack action $\mathbf{A}_t$ at t
Sample a new graph state $\mathbf{S}_t$ at t
Update belief $b_t(\mathbf{S}_t)$

Figure 6: Overview of the defender belief update on graph states

at each time step. This approach allows us to limit the number of graph states and attack actions considered. The overview of the particle filtering method is shown in Figure 6.

Essentially, the method samples particles (each particle is a sample of the graph state at time step t). Given belief $\mathbf{b}_{t-1}$, which provides a probability distribution over graph state set $\mathfrak{S}(\mathbf{b}_{t-1})$, we randomly sample a graph state $\mathbf{S}_{t-1}$ at $t-1$ based on this distribution. Given the sampled graph state $\mathbf{S}_{t-1}$, we sample an attack action $\mathbf{A}_t$, according to $\mathbf{S}_{t-1}$ and the assumption of the defender about which level-0 heuristic strategy the attacker plays. We then sample a new graph state $\mathbf{S}_t$ at time step t based on: (i) sampled graph state $\mathbf{S}_{t-1}$; (ii) sampled attack action $\mathbf{A}_t$; and (iii) the defender's action $\mathbf{D}_t$. Finally, we update the defender's belief $b_t(\mathbf{S}_t)$ for the sampled graph state $\mathbf{S}_t$ based on the defender's observation $\mathbf{O}_t$ at time step t.

Finding an optimal defense action at $t+1$ given the defender's updated belief $b_t(\mathbf{S}_t)$ requires taking into account an assumption regarding the attacker's strategy and all possible future outcomes of the game as a result of the defender's action at $t+1$, which is computationally expensive. Therefore, we propose two different heuristic strategies for the defender. In the following section, we propose two new attack-modeling-based defense strategies that take into account the defender's belief, 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

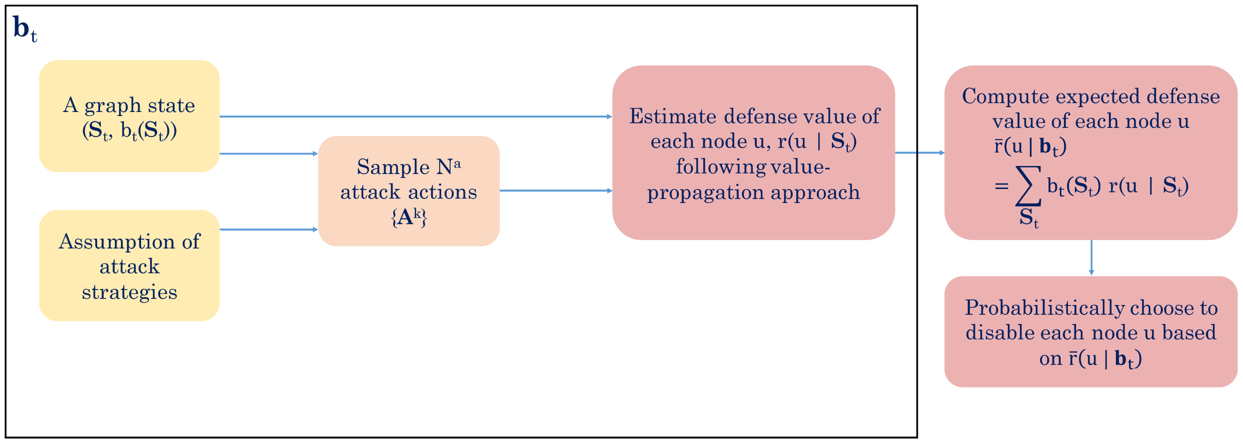
At each time step $t+1$, the defender has belief $\mathbf{b}_t$ on possible graph states at the end of time step t. For each state in the belief set $\mathbf{S}_t \in \mathfrak{S}(\mathbf{b}_t)$, we define the defense candidate set, $\mathbf{\Psi}^d(\mathbf{S}_t)$, which consists of: active goal nodes,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

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

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

5.4 Level-1 value-propagation defense strategy

The overview of the level-1 value-propagation defense strategy is illustrated in Figure 7. For each graph state $\mathbf{S}_t$ in the belief set $\mathbf{S}_t \in \mathfrak{S}(\mathbf{b}_t)$, based on the assumption of which heuristic strategy is played by the attacker, the defense strategy first samples N^a attack actions $\{\mathbf{A}^k\}$ for $k \in \{1, 2, \ldots, N^a\}$. The strategy then estimates the defense value $r(u \mid \mathbf{S}_t)$ of each node u in the defense set $u \in \mathbf{\Psi}^d(\mathbf{S}_t)$ based on the graph state $\mathbf{S}_t$ and the set of sampled attack actions $\{\mathbf{A}^k\}$ following the value-propagation approach, which will be explain later. This process is repeated for all graph states in the belief set $\mathfrak{S}(\mathbf{b}_t)$. Finally, the value-propagation defense strategy computes the expected defense value $\bar{r}(u)$ for candidate nodes $u \in \cup_{\mathbf{S}_t} \mathbf{\Psi}^d(\mathbf{S}_t)$ based on the defender's belief $\mathbf{b}_t$ and then probabilistically selects nodes to disable based on this expected defense value.



b_t
A graph state (S_t, b_t(S_t))
Assumption of attack strategies
Sample N^a attack actions {A^k}
Estimate defense value of each node u, r(u | S_t) following value-propagation approach
Compute expected defense value of each node u r̄(u | b_t) = Σ_{S_t} b_t(S_t) r(u | S_t)
Probabilistically choose to disable each node u based on r̄(u | b_t)

Figure 7: Overview of the value-propagation defense strategy

For each possible graph state $\mathbf{S}_t \in \mathfrak{S}(\mathbf{b}_t)$, we first estimate the propagated defense reward $\hat{r}(u \mid \mathbf{S}_t)$ of each node $u \in \mathbf{V}$ by propagating the defender's rewards $r^d(w) < 0$ at inactive goal nodes $w \in \mathbf{V}^g \setminus \mathbf{S}_t$ 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 complete procedure is presented as Algorithm 4. The idea of computing propagated defense rewards is similar to computing propagated attack values. In lines 3–12 of Algorithm 4, we compute the propagated defense reward $\hat{r}(u \mid \mathbf{S}_t)$ 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 $t' + 1$.

Based on computed propagated defense rewards, given the set of sampled attack actions $\{\mathbf{A}^k\}$, we then can estimate the defense values, $r(u \mid \mathbf{S}_t)$, for defense candidate nodes $u \in \mathbf{\Psi}^d(\mathbf{S}_t)$ as follows:

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

In particular, for active goal nodes $u \in \mathbf{V}^g \cap \mathbf{S}_t$, $r(u \mid \mathbf{S}_t)$ comprises not only the cost $c^d(u)$ but also the propagated defense reward $\hat{r}(u \mid \mathbf{S}_t)$ and the defender's reward, $r^d(u)$, at u. For other defense candidate nodes, $r(u \mid \mathbf{S}_t)$ 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 the probability a candidate node u becomes active, the higher the defense value for the defender to disable that node. The defense value for each node u (which is not an active goal node) takes into account the probability u becomes active (as a result of sampled attack actions), denoted by $p\left(s_{t+1}(u) = 1 \mid \mathbf{A}^k, \mathbf{S}_t\right)$, which is equal to:

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

where $I(\Phi)$ is a binary indicator for condition Φ. Finally, the probability the defender will choose each node u to disable is computed according to the conditional logistic function of the expected defense values. The number of chosen nodes to disable is a certain fraction of the cardinality of $\cup_{\mathbf{S}_t} \mathbf{\Psi}^d(\mathbf{S}_t)$.

Algorithm 4: Compute Propagated Defense Reward

```
1  Input: time step, $t+1$, graph state, $\mathbf{S}_t$, 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 $w \in \mathbf{V}^g \setminus \mathbf{S}_t$, for all time steps $t' \in \{t+1,\ldots,T\}$;
3  for $u \in itopo(\mathbf{G})$ do
4      for $v \in \pi^+(u) \setminus \mathbf{S}_t$ do
5          for $w \in \mathbf{V}^g \setminus (\mathbf{S}_t \cup \{u\}), t' \in \{t+1 \ldots T-1\}$ do
6              if $v \in \mathbf{V}^\wedge$ then
7                  $r^w(v \to u, t'+1) \leftarrow p(v) r^w(v,t')$;
8              else
9                  $r^w(v \to u, t'+1) \leftarrow p(u,v) r^w(v,t')$;
10             if $r^w(u,t'+1) > \gamma r^w(v \to u, t'+1)$ then
11                 Update $r^w(u,t'+1) \leftarrow \gamma r^w(v \to u, t'+1)$;
12 Return $\hat{r}(u \mid \mathbf{S}_t) = \min_{w \in \mathbf{V}^g \setminus \mathbf{S}_t} \min_{t' \in \{t+1,\ldots,T\}} r^w(u,t') \, (\neq +\infty), \forall u$;
```

5.5 Level-1 sampled-activation defense strategy

In this defense strategy, we leverage the random activation process as described in Section 4.2.4 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. The defense value of each defense action indicates the impact of the defender's action on preventing the attacker from attacking goal nodes. We then select the action which leads to the highest defense value. An overview of the sampled-activation defense strategy is illustrated in Figure 8. The details of the strategy are explained in the following.

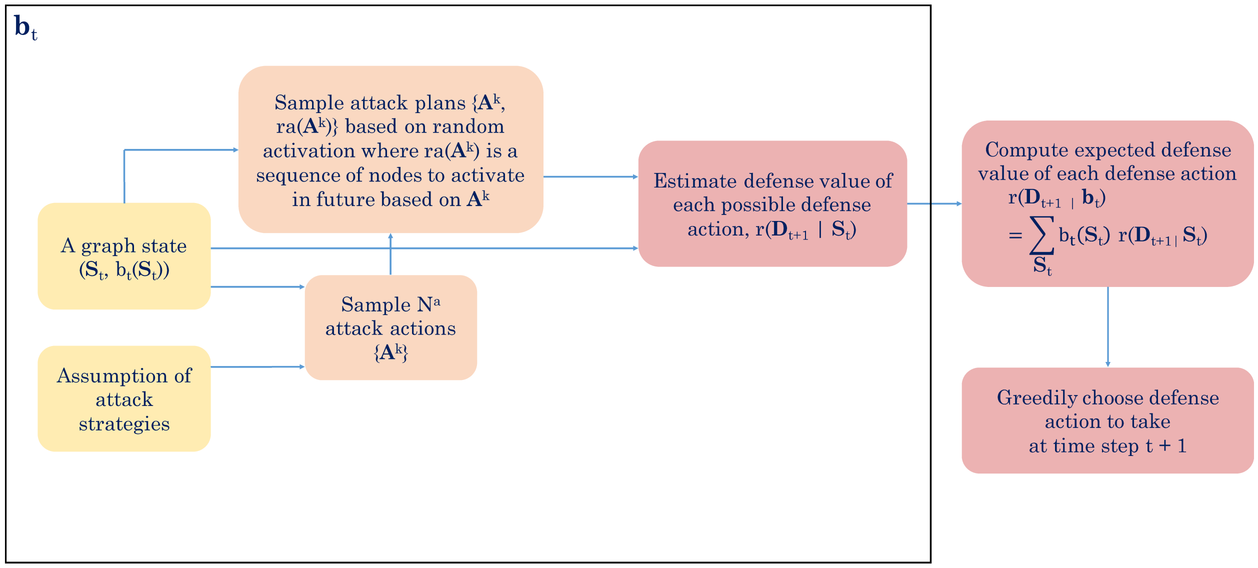

$\mathbf{b}_t$
Sample attack plans $\{\mathbf{A}^k, \mathrm{ra}(\mathbf{A}^k)\}$ based on random activation where $\mathrm{ra}(\mathbf{A}^k)$ is a sequence of nodes to activate in future based on $\mathbf{A}^k$
A graph state $(\mathbf{S}_t, b_t(\mathbf{S}_t))$
Sample N^a attack actions $\{\mathbf{A}^k\}$
Assumption of attack strategies
Estimate defense value of each possible defense action, $r(\mathbf{D}_{t+1} \mid \mathbf{S}_t)$
Compute expected defense value of each defense action $r(\mathbf{D}_{t+1} \mid \mathbf{b}_t) = \sum_{\mathbf{S}_t} b_t(\mathbf{S}_t)\, r(\mathbf{D}_{t+1} \mid \mathbf{S}_t)$
Greedily choose defense action to take at time step t + 1

Figure 8: Overview of the sampled-activation defense strategy

Algorithm 5: Find blocked nodes

```
1  Input: $\mathbf{S}_t, \mathbf{D}_{t+1}, \mathbf{A}^k, ra(\mathbf{A}^k)$;
2  Initialize block status $isBlocked(v) \leftarrow 0$ for all $v \in ra(\mathbf{A}^k) \cup \mathbf{S}_t \setminus \mathbf{D}_{t+1}$ and $isBlocked(v) \leftarrow 1$ for all
   $v \in \mathbf{D}_{t+1}$;
3  for $u \in ra(\mathbf{A}^k) \setminus \mathbf{V}^r$ with $isBlocked(u) = 0$ do
4      if $u \in \mathbf{V}^\vee$ then
5          if $isBlocked(pre(u))$ then
6              $isBlocked(u) \leftarrow 1$;
7      else
8          if $isBlocked(v)$ for some $v \in pre(u)$ then
9              $isBlocked(u) \leftarrow 1$;
10 Return $\{isBlocked(u)\}$.
```

5.5.1 Sample attack plans.

We first sample attack plans of the attacker at time step $t+1$. Each plan refers to a pair of an attack action at $t+1$ and a corresponding attack path the attacker may take to reach inactive goal nodes in future. 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 of the subset of inactive goal nodes with the highest attack value according to that random activation (Section 4.2.4). 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.

For each possible graph state $\mathbf{S}_t \in \mathfrak{S}(\mathbf{b}_t)$, 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 $\mathbf{D}_{t+1} \subseteq \mathbf{\Psi}^d(\mathbf{S}_t)$ 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 $v \in \mathbf{V}^g \cap ra(\mathbf{A}^k)$ 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 5, 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:

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

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 which either are active or will be activated based on the attack plan $(\mathbf{A}^k, ra(\mathbf{A}^k))$. Finally, based on the defense value $r(\mathbf{D}_{t+1} \mid \mathbf{S}_t)$ with respect to the graph state $\mathbf{S}_t$, the expected defense value of each $\mathbf{D}_{t+1}$ over the defender's belief is computed as follows:

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

Proposition 5. *Given a graph state $\mathbf{S}_t \in \mathfrak{S}(\mathbf{b}_t)$, a defense action $\mathbf{D}_{t+1}$, and a set of N^a sampled attack plans $\{(\mathbf{A}^k, ra(\mathbf{A}^k))\}$, the time complexity of the estimation of the defense value of $\mathbf{D}_{t+1}$ is $O\left((|\mathbf{V}| + |\mathbf{E}|) \times N^a\right)$.*

Proof. Given a graph state $\mathbf{S}_t \in \mathfrak{S}(\mathbf{b}_t)$, a defense action $\mathbf{D}_{t+1}$, and a sampled attack plan $(\mathbf{A}^k, ra(\mathbf{A}^k))$, Algorithm 5 iterates over each node u in the attack sequence $ra(\mathbf{A}^k)$ to update the blocked status of u based on the preconditions of this node. Because of the directed acyclic topological structure of the graph, Algorithm 5 has to examine each node in the sequence and its incoming edges once. Therefore, the time complexity of Algorithm 5 is $O(|\mathbf{V}| + |\mathbf{E}|)$. In addition, given the blocked status of the graph computed by Algorithm 5, we estimate the defense value $r(\mathbf{D}_{t+1} \mid \mathbf{S}_t, \mathbf{A}^k, ra(\mathbf{A}^k))$ based on the cost of disabling nodes in $\mathbf{D}_{t+1}$ and the potential loss regarding goal nodes of which attack paths are not blocked. The time complexity of this estimation is $O(|\mathbf{V}|)$ since $|\mathbf{D}_{t+1}| + |\mathbf{V}^g| \leq 2 \times |\mathbf{V}|$. Since there are N^a sampled attack plans, the time complexity of computing the defense value $r(\mathbf{D}_{t+1} \mid \mathbf{S}_t)$ is $O\left((|\mathbf{V}| + |\mathbf{E}|) \times N^a\right)$. □

5.5.3 Greedy defense strategy.

Finding an optimal defense action according to the expected defense value $r(\mathbf{D}_{t+1} \mid \mathbf{b}_t)$ 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 $\mathbf{b}_t$. These nodes are chosen from the defense candidate set with respect to the defender's belief $\mathbf{b}_t$, which is defined as $\cup_{\mathbf{S}_t \in \mathfrak{S}(\mathbf{b}_t)} \mathbf{\Psi}^d(\mathbf{S}_t)$ where $\mathbf{\Psi}^d(\mathbf{S}_t)$ is the defense candidate set with respect to the graph state $\mathbf{S}_t$. 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: $r(\mathbf{D}_{t+1} \cup \{u\} \mid \mathbf{b}_t) - r(\mathbf{D}_{t+1} \mid \mathbf{b}_t) \leq 0$ for all u.

Proposition 6. *Suppose that the number of graph states to sample in particle filtering is N^s (which means that $|\mathfrak{S}(\mathbf{b}_t)| = N^s$). The time complexity of the static greedy heuristic is $O(|\mathbf{V}|^2 \times N^s \times (|\mathbf{V}| + |\mathbf{E}|) \times N^a)$.*

Proof. The size of the defense candidate set $\cup_{\mathbf{S}_t \in \mathfrak{S}(\mathbf{b}_t)} \mathbf{\Psi}^d(\mathbf{S}_t)$ is at most $|\mathbf{V}|$. Therefore, there are at most $|\mathbf{V}|$ iterations of finding next best node in the static greedy heuristic. In each iteration of the heuristic, given the current set of selected nodes $\mathbf{D}_{t+1}$, for each candidate node u, we compute the defense value $r(\mathbf{D}_{t+1} \cup \{u\} \mid \mathbf{b}_t)$, this step takes $O(N^s \times (|\mathbf{V}| + |\mathbf{E}|) \times N^a)$ time according to Proposition 5. Since there are at most $|\mathbf{V}|$ nodes in the candidate set, the time complexity of static greedy heuristic is $O(|\mathbf{V}|^2 \times N^s \times (|\mathbf{V}| + |\mathbf{E}|) \times N^a)$. □

In addition, we propose a randomized greedy heuristic to select a defense action from a set of greedy defense actions probabilistically. These greedy defense actions are generated based on each $\mathbf{S}_t$ in $\mathfrak{S}(\mathbf{b}_t)$.

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 its choice over $\{\mathbf{D}_{t+1}(\mathbf{S}_t)\}$ according to a conditional logistic distribution with respect to the defense value $\{r(\mathbf{D}_{t+1}(\mathbf{S}_t) \mid \mathbf{b}_t)\}$ computed based on the defender belief $\mathbf{b}_t$. The following proposition provides the time complexity of the randomized greedy heuristic. This proposition can be proved in a similar way as the complexity of the static greedy heuristic.

Proposition 7. *The time complexity of the randomized greedy heuristic is $O(|\mathbf{V}|^2 \times N^s \times (|\mathbf{V}| + |\mathbf{E}|) \times N^a)$.*

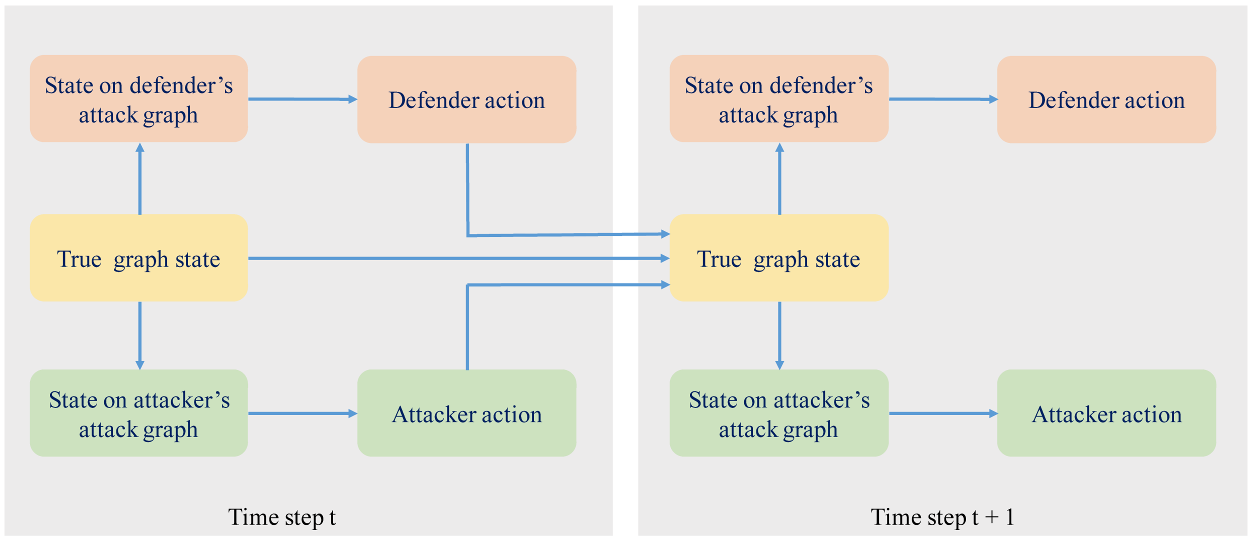

State on defender's attack graph
Defender action
True graph state
State on attacker's attack graph
Attacker action
Time step t
State on defender's attack graph
Defender action
True graph state
State on attacker's attack graph
Attacker action
Time step t + 1

Figure 9: Empirical game-theoretic analysis simulation

6 Incorporating Uncertainty about the Attack Graph

In the previous sections, we presented our security game model and heuristic strategies of players on the same attack graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ – both the defender and the attacker know the true underlying attack graph of the network system. In this section, we discuss an extension of our model and algorithms to incorporate uncertainty about the attack graph from the players' perspectives. In other words, both the defender and the attacker have their own attack graphs which are different from the true one.

In particular, we denote by $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ the true attack graph of the network system. Let $\mathbf{G}^a = (\mathbf{V}^a, \mathbf{E}^a)$ and $\mathbf{G}^d = (\mathbf{V}^d, \mathbf{E}^d)$ be the attack graphs of the attacker and defender respectively. Here, $\mathbf{V}^a \neq \mathbf{V}^d (\neq \mathbf{V})$ and $\mathbf{E}^a \neq \mathbf{E}^d (\neq \mathbf{E})$ are the sets of nodes and exploits from the view of the attacker and the defender of the network system. Given this extension, it is straightforward to redefine our heuristic strategies for the attacker and the defender based on $\mathbf{G}^a$ and $\mathbf{G}^d$ separately. For example, the *uniform* attack heuristic chooses an attack action from the exploit set $\mathbf{E}^a$ uniformly at random. Similarly, the *value-propagation* attack heuristic performs the attack value propagation as well as the probabilistic selection of an attack action based on the attack graph $\mathbf{G}^a = (\mathbf{V}^a, \mathbf{E}^a)$ instead of the true $\mathbf{G} = (\mathbf{V}, \mathbf{E})$.

In evaluating the effectiveness of our proposed heuristic strategies, our empirical game-theoretic analysis simulates the attack progression on all three attack graphs: (i) the true attack graph; (ii) the defender's attack graph; and (iii) the attacker's attack graph. The whole simulation process is illustrated in Figure 9. The states of the attack graphs with respect to the defender and the attacker are determined based on the states of the true attack graphs. The transition of the states on the true attack graphs depends on the actions of the players chosen from our heuristic strategies. For example, if the attacker performs a *non-existing* exploit, there will be no change of the true graph state. Finally, players' payoffs are defined based on the true attack graph states, which are used to evaluate the effectiveness of the heuristic strategies.

7 Experiments

We evaluate the effectiveness of the proposed strategies in various settings with different graph topologies, node type ratios, and levels of defender observation noise. As in prior EGTA treatments of cybersecurity scenarios [Wright et al., 2016], we employ the simulation data to estimate a two-player normal-form game model of the strategic interaction.

7.1 Player strategies

We tune the strategies for the players by adjusting their parameter values. In our experiments, the fraction 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 fraction 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 No-op (*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 $\eta^a \in \{1.0, 3.0\}$.

7.2 Simulation settings

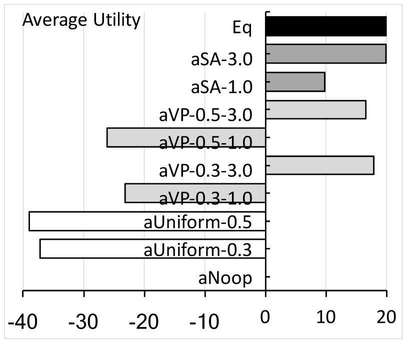
We consider two types of graph topology: (i) layered directed acyclic graphs (layered DAGs); and (ii) random directed acyclic graphs (random DAGs). Graphs in the former case consist of multiple separate layers with edges connecting only nodes in consecutive layers. We generate 5-layered DAGs. The k^{th} 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 (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 proportion of $\wedge$-nodes is either zero or half.

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.

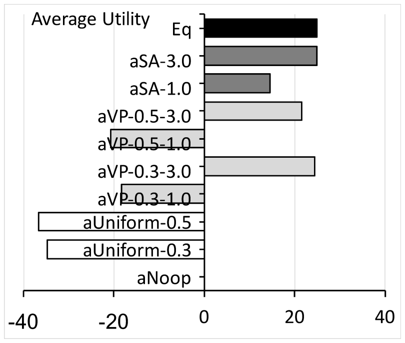
We consider three cases of observation noise levels: (i) high noise: the signal probabilities $p(1_v \mid s(v) = 1)$ and $p(1_v \mid s(v) = 0)$ are generated within $[0.6, 0.8]$ and $[0.2, 0.4]$ uniformly at random respectively; (ii) low noise: $p(1_v \mid s(v) = 1)$ and $p(1_v \mid s(v) = 0)$ are generated 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 $\gamma = 0.9$.

7.3 Strategy comparison

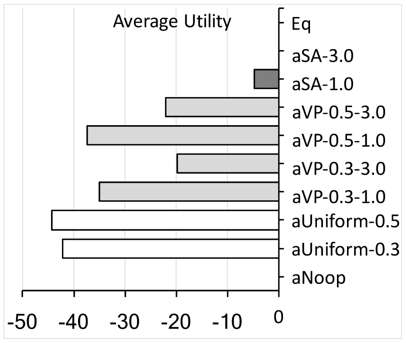
Based on the aforementioned settings, we generated 10 different games in each of our experiments. For each game, we ran 500 simulations to estimate the payoff of each pair of players' strategy instances. As a result, we obtain a payoff matrix for each game based on which we can compute Nash equilibria using Gambit [McKelvey et al., 2006]. 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.



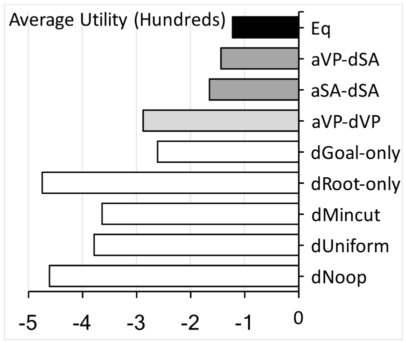
(a) Attacker, high noise



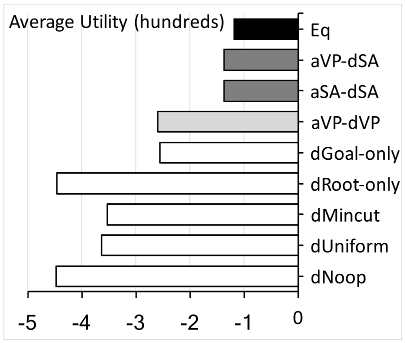
(b) Attacker, low noise



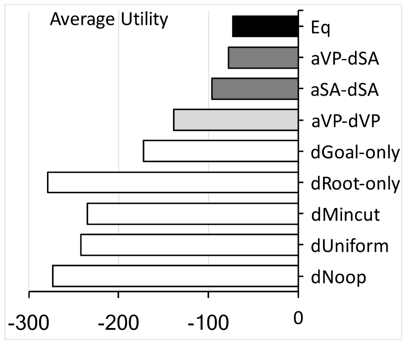
(c) Attacker, no noise



(d) Defender, high noise



(e) Defender, low noise



(f) Defender, no noise

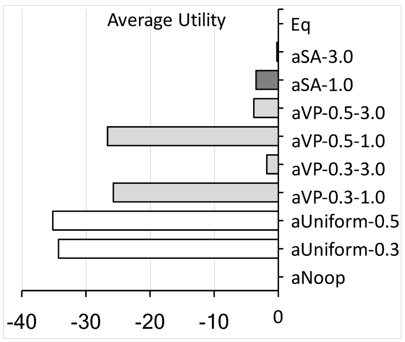
Figure 10: Strategy evaluation, layered DAGs, 0% $\wedge$-nodes

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.

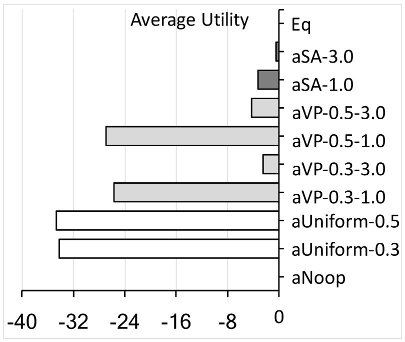
7.3.1 Results on layered DAGs

Our first set of experiments is based on layered DAGs, as shown in Figure 10 (0% $\wedge$-nodes) and Figure 11 (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 strategy instances played by the players. For the purpose of analysis, we group these strategy instances based on heuristic strategies played, which are represented via the shading on bars. For example, Figures 10(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%.

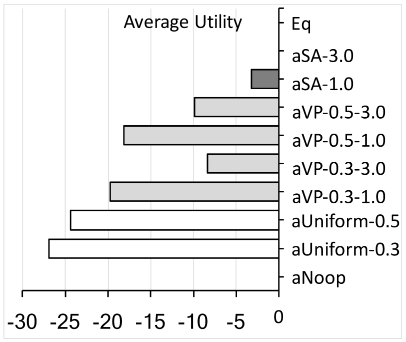
In Figures 10(d)(e)(f), and 11(d)(e)(f), the defense sampled-activation strategy (aVP-dSA and aSA-dSA) obtains the defense utility closest to the equilibrium utility (Eq) in all game settings regardless of the assumed attack strategies (i.e., whether the attacker follows the value-propagation or sampled-activation attack strategies with certain parameter values). In fact, when the percentage of $\wedge$-nodes is 0%, the Nash equilibria obtained for all games based on EGTA comprise only 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 do not suffer from a significant loss in utility. This result implies that the sampled-activation defense strategies are also robust to the defender's uncertainty about true graph state. 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



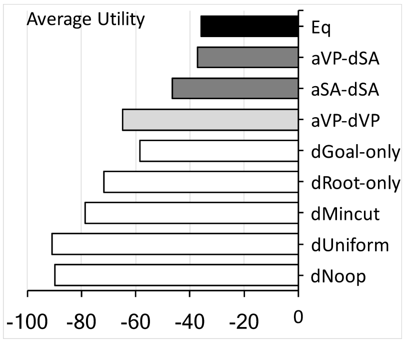
(a) Attacker, high noise



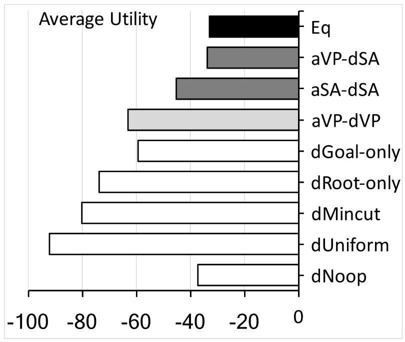
(b) Attacker, low noise



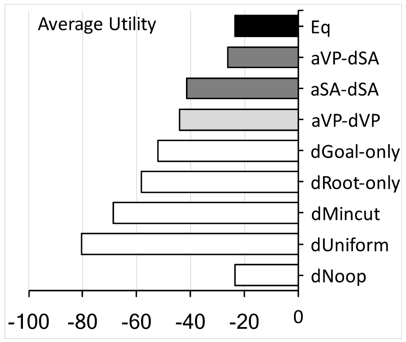
(c) Attacker, no noise



(d) Defender, high noise



(e) Defender, low noise



(f) Defender, no noise

Figure 11: Strategy evaluation, layered DAGs, 50% $\wedge$-nodes

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 10(a)(b)(c) and 11(a)(b)(c) show that in all game settings, the attacker 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 $\wedge$-nodes is 50%, aNoop gets a high probability in the attacker's equilibrium strategies in all game settings. In Figures 11(a)(b)(c), aNoop obtains an attacker utility (which is zero) approximately the same as the equilibrium attack strategy. In fact, when the number of $\wedge$-nodes is large, it is difficult for the attacker to intrude deeply in the attack graph, because compromising $\wedge$-nodes takes more effort. Consequently, the defender obtains a significantly higher utility when the percentage of $\wedge$-nodes is 50% (Figures 11(d)(e)(f)) than when it is 0% (Figures 10(d)(e)(f)). The dNoop strategy is also included in the defender's equilibrium strategy when the $\wedge$-node percentage is 50%. Finally, in Figure 10(c), the attacker's equilibrium utility is approximately zero even when all the nodes are of $\vee$-type. This result shows that when the defender knows the graph state, our sampled-activation defense strategy is highly effective such that the attacker cannot achieve any benefit from attacking.

In addition to the results on average in Figure 10 and Figure 11, we provide a detailed equilibrium analysis of each individual game. We provide detailed equilibrium results of each individual game with respect to layered directed acyclic graphs with the percentage of $\wedge$-nodes is 0%. In particular, Figures 14&15 show equilibria of 10 games in the case of high noise. Figure 16&17 show equilibria of 10 games in the case of low noise. Finally, Figure 18&19 show equilibria of 10 games in the case of no noise. In addition to the results on the attacker and the defender's expected utility, we present the equilibrium of each game which consists of the probability the players play each strategy instance. For example, Figure 14(a) shows the equilibrium analysis of Game 1. In Game 1, the defender's equilibrium strategy is to play (i) the strategy instance (aVP-0.5-3.0)-(dSA-3.0-ranG) (the defender plays the sampled-activation defense strategy following the randomized greedy heuristic with $\eta^d = 3.0$, assuming the attacker plays the value propagation strategy with $\eta^a = 3.0$ and $p^a = 0.5$) with a probability of 0.943; and (ii) the strategy instance (aVP-0.3-3.0)-(dSA-3.0-ranG) (the defender plays the sampled-activation defense strategy following the randomized greedy heuristic with $\eta^d = 3.0$, assuming the attacker plays the value propagation strategy with $\eta^a = 3.0$ and $p^a = 0.3$) with a probability of 0.943. On average over all strategy instances according to different parameter values, the defender's strategy aVP-dSA obtains the expected utility of the defender closest to the equilibrium utility.

Overall, the strategy instances of the players involved in each game's equilibrium vary across different games. Yet, the result of the players' expected utility for playing each heuristic strategy (averaged over all corresponding strategy instances) is consistent among all games. For example, the sampled-activation defense strategy outperforms other heuristic defense strategies in terms of obtaining an expected utility of the defender closest to the equilibrium utility in all games.

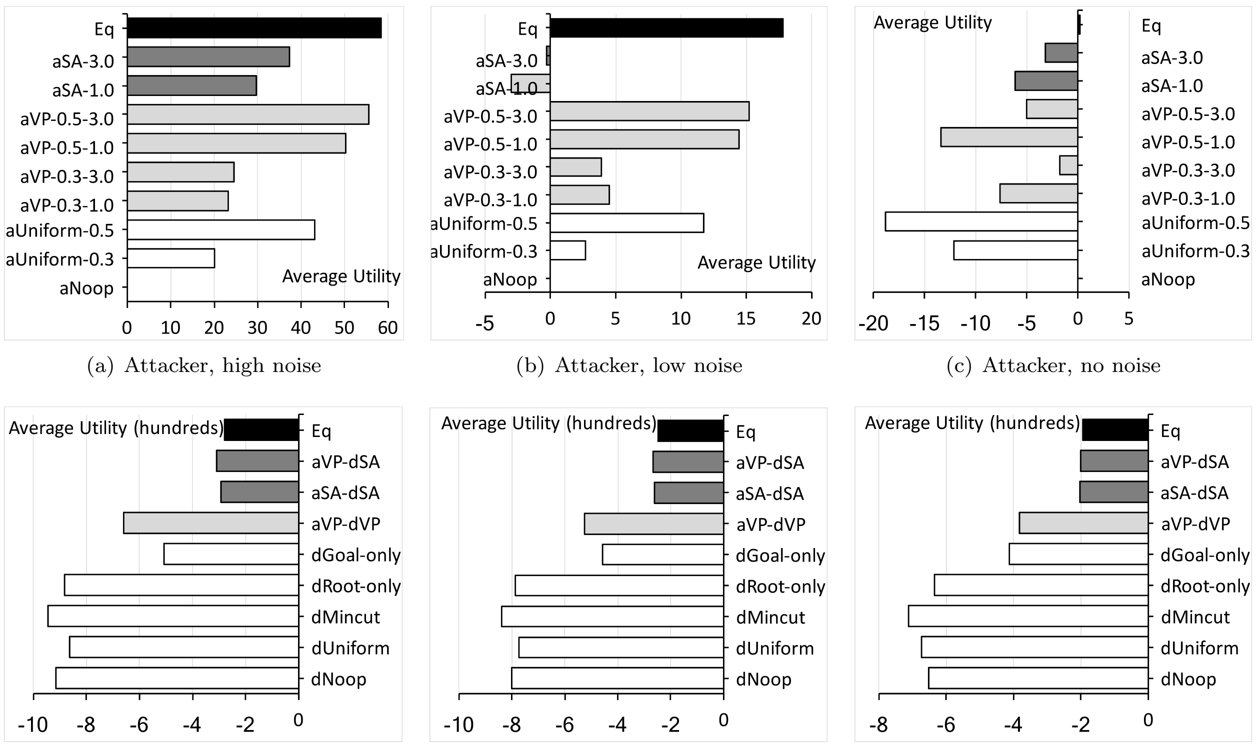
7.3.2 Results on random DAGs

Our second set of experiments is based on random DAGs (Figure 12). This figure shows that the solution quality of the defender's strategies is 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 12(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 12(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 12(c)), aVP-0.5-3.0 and aVP-0.3-3.0 with $\eta^a = 3.0$ obtains 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 12(b)), aVP-0.5-3.0 and aVP-0.5-1.0 with $per^a = 0.5$ obtains the highest attack utility compared with other attack strategies.

7.4 Strategy robustness

In our third set of experiments, we evaluate the robustness of the defender's strategies to the uncertainty about the attacker's strategy. In particular, in addition to the nine aforementioned attack strategy instances, we consider seven more strategy instances corresponding to new parameter values: $p^a = 0.1$ and $\eta^a = 5.0$. The seven new attack strategy instances include: (i) one Uniform instance with $p^a = 0.1$; (ii) four Value-propagation instances with $p^a = 0.1$ and $\eta^a = 5.0$; and (iii) two Sampled-activation instances with $\eta^a = 5.0$. Meanwhile, we use the same 43 defense strategy instances as in the previous sets of experiments. None of these defense strategy instances is based on a model of the attacker using any of the seven new attack strategy instances. We use Nash equilibria of the resulting new games to analyze the solution quality of the proposed strategies. We aim to examine both the attacker's gain and the defender's loss in utility with respect to the new attack strategy instances. The experimental results are shown in Figure 13 for layered DAGs with 0% $\wedge$-nodes.

Figures 13(a)(b)(c) show that the attacker obtains a significantly higher utility on average compared to the corresponding results shown in Figure 10(a)(b)(c). On the other hand, the defender suffers a moderate loss in utility (Figure 10(d)(e)(f) versus Figures 13(d)(e)(f)). For example, in the case of high noise in the defender's observations, the attacker utility is 20.0, at equilibrium over the original nine attack strategy instances (Figure 10(a)). The attacker's equilibrium utility is 55.0 (2.75 times higher) when the seven new



(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 12: Strategy evaluation, random DAGs, 0% $\wedge$-nodes

attack strategy instances are also allowed (Figure 13(a)). Conversely, the defender's equilibrium utility is approximately -177.0 with the seven new attacker strategies allowed (Figure 13(d)), which is 1.45 times lower than its equilibrium utility of -122.0 for the original, smaller attacker strategy set (Figure 10(d)).

The new attacker strategy instances yield significantly higher utility for the attacker than the original set. For example, the sampled-activation instance aSA-5.0 with $\eta^a = 5.0$ obtains a utility of 32.87 (Figure 13(a)), while aSA-3.0 and aSA-1.0 with $\eta^a = 3.0$ and $\eta^a = 1.0$ only get a utility of 20.43 and 10.67 respectively. Figure 13(a) shows that the attacker's utility achieved by the value-propagation strategy instances with new $\eta^a = 5.0$, including aVP-0.5-5.0, aVP-0.3-5.0, and aVP-0.1-5.0, is closer to the equilibrium utility than the utility of the earlier value-propagation instances.

Although the defender suffers reduced payoffs when the new attack strategy instances are allowed, the relative performance of the defender's heuristic strategies are consistent with the previous experimental results (Figure 10(d)(e)(f) versus Figures 13(d)(e)(f)). In particular, the sampled-activation defense strategy still obtains the highest utility (which is close to the equilibrium utility) for the defender, regardless of which assumption the defender makes about the attacker's strategy. Among the graph-based defense strategies, the dGoal-only strategy performs the best in all three cases of the defender's observation noise level.

8 Summary

We study the problem of deploying security countermeasures on Bayesian attack graphs to protect networks 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

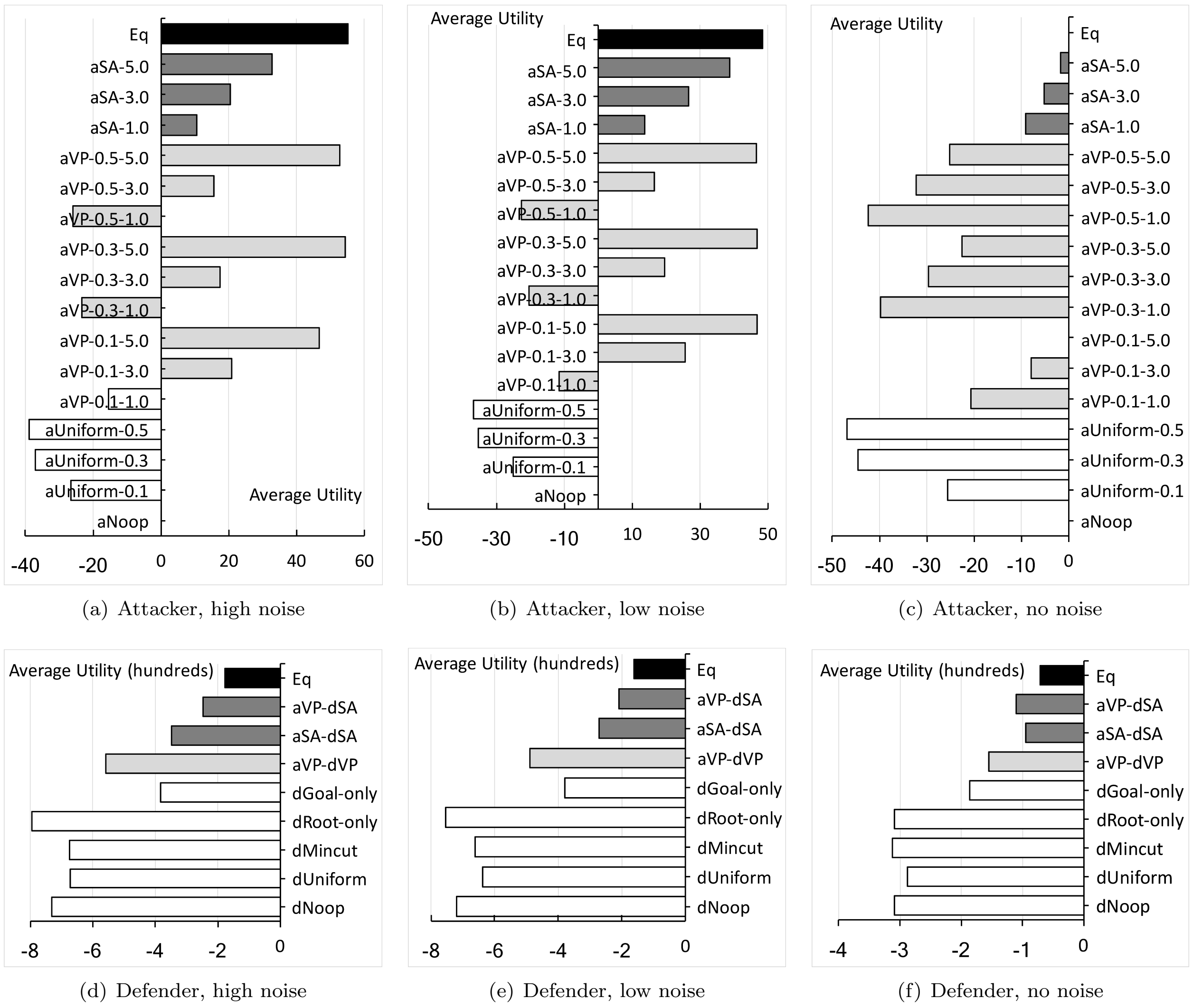


Figure 13: Robustness evaluation, layered DAGs, 0% ∧-nodes

an analytical 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. We provide a detailed analysis of the time complexity of our proposed heuristic strategies, which is polynomial. 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 the security environment, including 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. We benefited greatly from discussions with colleagues on this project, especially Erik Miehling and Massimiliano Albanese.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

References

Massimiliano Albanese and Sushil Jajodia. A graphical model to assess the impact of multi-step attacks. *Journal of Defense Modeling and Simulation*, 15:79–93, 2018.

Tansu Alpcan and Tamer Basar. A game theoretic analysis of intrusion detection in access control systems. In *Proc. of the 43rd IEEE Conference on Decision and Control*, pages 1568–1573, 2004.

Tansu Alpcan and Tamer Basar. An intrusion detection game with limited observations. In *Proceedings of the 12th Int. Symp. on Dynamic Games and Applications*, 2006.

Stefano Bistarelli, Marco Dall'Aglio, and Pamela Peretti. Strategic games on defense trees. In *International Workshop on Formal Aspects in Security and Trust*, pages 1–15. Springer, 2006.

Colin F. Camerer, Teck-Hua Ho, and Juin-Kuan Chong. A cognitive hierarchy model of games. *Quarterly Journal of Economics*, 119:861–898, 2004.

M. Dacier, Y. Deswarte, and M. Kaâniche. Models and tools for quantitative assessment of operational security. In Sokratis K. Katsikas and Dimitris Gritzalis, editors, *Information Systems Security*, pages 177–186. Springer, 1996.

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

Suguo Du, Xiaolong Li, Junbo Du, and Haojin 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.

Karel Durkota, Viliam Lisỳ, Branislav Bošanskỳ, and Christopher Kiekintveld. Approximate solutions for attack graph games with imperfect information. In *6th International Conference on Decision and Game Theory for Security*, pages 228–249, 2015a.

Karel Durkota, Viliam Lisỳ, Branislav Bošanskỳ, and Christopher Kiekintveld. Optimal network security hardening using attack graph games. In *24th International Joint Conference on Artificial Intelligence*, pages 526–532, 2015b.

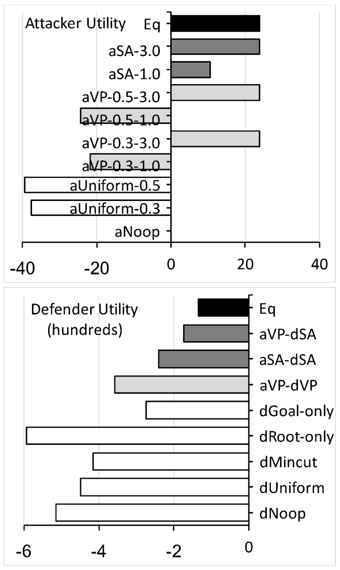
David Evans, Anh Nguyen-Tuong, and John Knight. Effectiveness of moving target defenses. In Jajodia et al. [2011].

Marcel Frigault, Lingyu Wang, Anoop Singhal, and Sushil Jajodia. Measuring network security using dynamic Bayesian network. In *4th ACM Workshop on Quality of Protection*, pages 23–30, 2008.

Kyle Ingols, Matthew Chu, Richard Lippmann, Seth Webster, and Stephen Boyer. Modeling modern network attacks and countermeasures using attack graphs. In *Computer Security Applications Conference, 2009. ACSAC'09. Annual*, pages 117–126. IEEE, 2009.

Sushil Jajodia, Anup K. Ghosh, Vipin Swarup, Cliff Wang, and X. Sean Wang, editors. *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*. Springer, 2011.

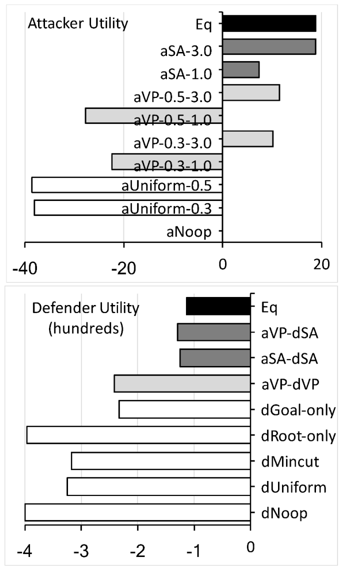
Barbara Kordy, Sjouke Mauw, Matthijs Melissen, and Patrick Schweitzer. Attack–defense trees and two-player binary zero-sum extensive form games are equivalent. In *1st International Conference on Decision and Game Theory for Security*, pages 245–256. Springer, 2010.



Defender	(aVP-0.5-3.0)-(dSA-3.0-ranG)	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	0.943	0.057

Attacker	aSA-3.0	aVP-0.3-3.0	aVP-0.5-3.0
Equilibrium	0.283	0.036	0.680

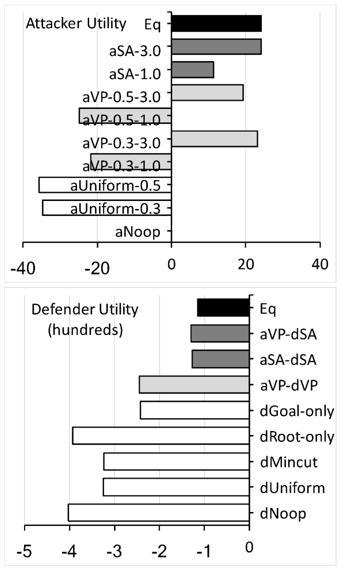
(a) Game 1



Defender	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	1.000

Attacker	aSA-3.0
Equilibrium	1.000

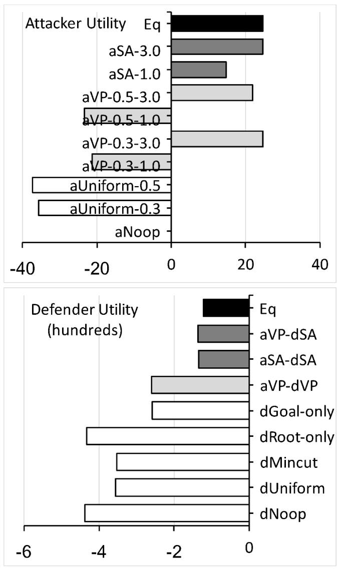
(b) Game 2



Defender	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	1.000

Attacker	aSA-3.0
Equilibrium	1.000

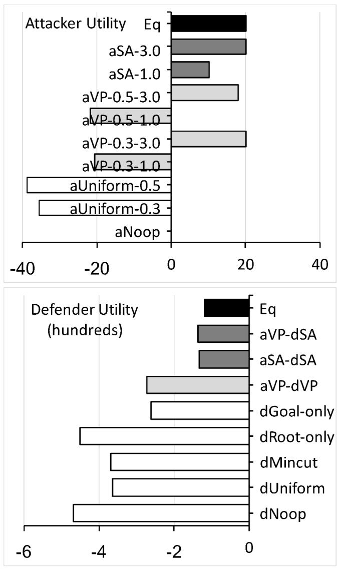
(c) Game 3



Defender	(aSA-3.0)-(dSA-1.0-staG)	(aVP-0.5-3.0)-(dSA-3.0-ranG)
Equilibrium	0.011	0.989

Attacker	aSA-3.0	aVP-0.3-3.0
Equilibrium	0.998	0.002

(d) Game 4

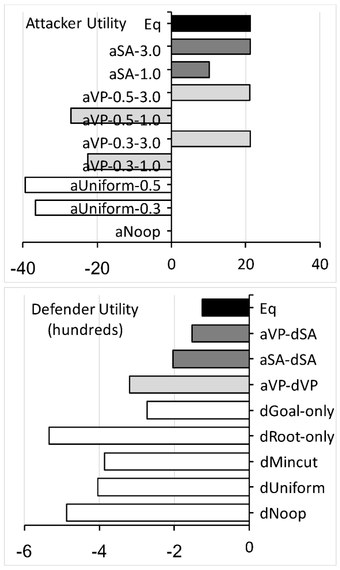


Defender	(aSA-3.0)-(dSA-3.0-staG)	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	0.013	0.987

Attacker	aSA-3.0	aVP-0.3-3.0
Equilibrium	0.985	0.015

(e) Game 5

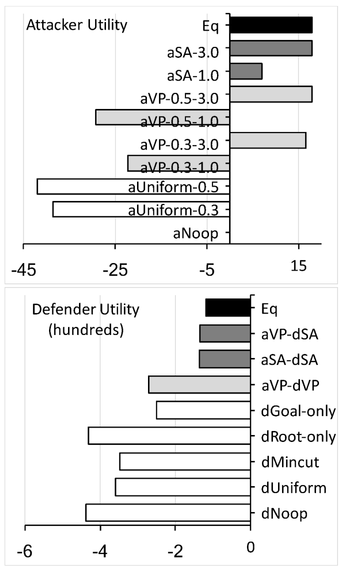
Figure 14: Equilibrium analysis, layered DAGs, 0% $\wedge$-nodes, high noise, games 1–5



Defender	(aVP-0.5-3.0)-(dSA-3.0-ranG)	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	0.790	0.210

Attacker	aSA-3.0	aVP-0.3-3.0
Equilibrium	0.482	0.518

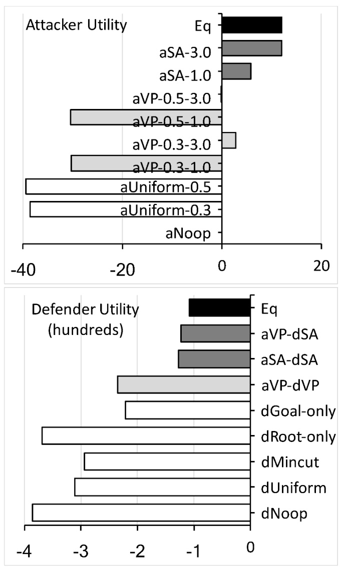
(a) Game 6



Defender	(aSA-1.0)-(dSA-3.0-staG)	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	0.050	0.950

Attacker	aSA-3.0	aVP-0.5-3.0
Equilibrium	0.972	0.028

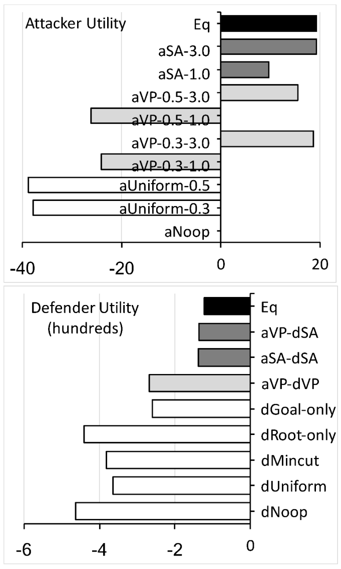
(b) Game 7



Defender	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	1.0

Attacker	aSA-3.0
Equilibrium	1.0

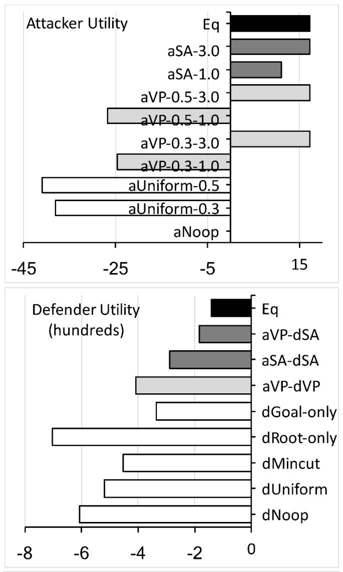
(c) Game 8



Defender	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	1.0

Attacker	aSA-3.0
Equilibrium	1.0

(d) Game 9

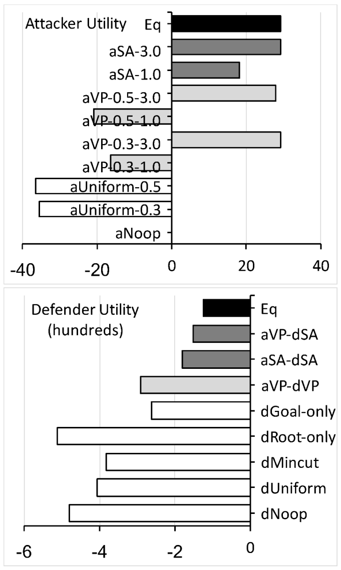


Defender	(aVP-0.5-3.0)-(dSA-3.0-ranG)	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	0.235	0.765

Attacker	aSA-3.0	aVP-0.3-3.0	aVP-0.5-3.0
Equilibrium	0.152	0.360	0.489

(e) Game 10

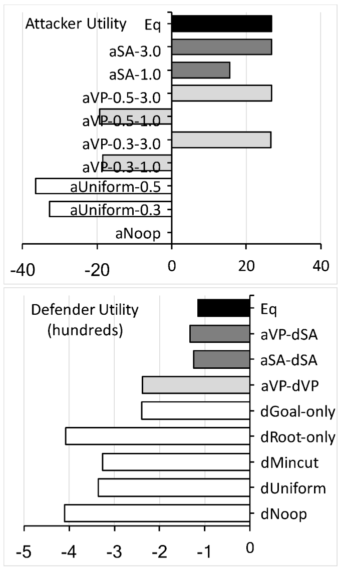
Figure 15: Equilibrium analysis, layered DAGs, 0% $\wedge$-nodes, high noise, games 6–10



Defender	(aVP-0.5-3.0)-(dSA-3.0-ranG)	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	0.615	0.385

Attacker	aSA-3.0	aVP-0.3-3.0
Equilibrium	0.594	0.406

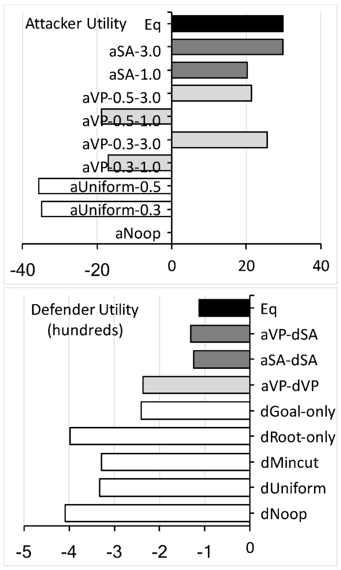
(a) Game 11



Defender	(aSA-1.0)-(dSA-3.0-staG)	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	0.070	0.930

Attacker	aSA-3.0	aVP-0.5-3.0
Equilibrium	0.947	0.053

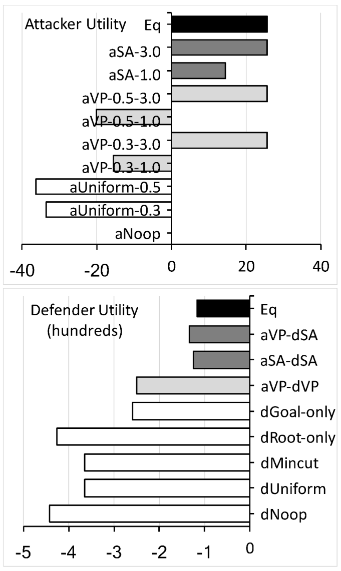
(b) Game 12



Defender	(aVP-0.5-3.0)-(dSA-3.0-ranG)
Equilibrium	1.00

Attacker	aSA-3.0
Equilibrium	1.00

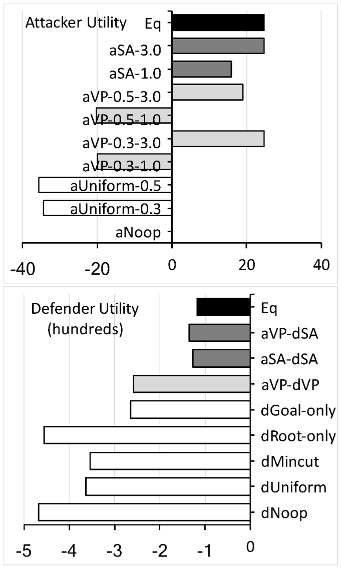
(c) Game 13



Defender	(aSA-1.0)-(dSA-1.0-staG)	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	0.068	0.932

Attacker	aSA-3.0	aVP-0.5-3.0
Equilibrium	0.975	0.025

(d) Game 14

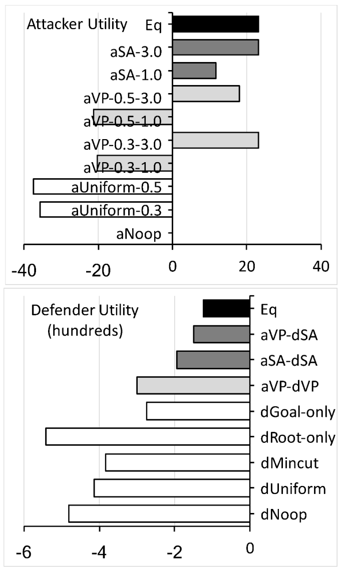


Defender	(aSA-3.0)-(dSA-3.0-staG)	(aVP-0.5-3.0)-(dSA-3.0-ranG)
Equilibrium	0.024	0.976

Attacker	aSA-3.0	aVP-0.3-3.0
Equilibrium	0.998	0.002

(e) Game 15

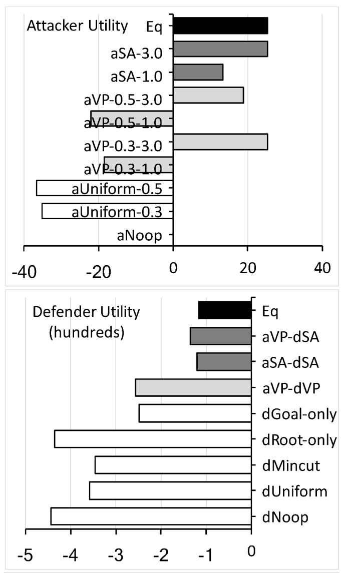
Figure 16: Equilibrium analysis, layered DAGs, 0% $\wedge$-nodes, low noise, games 11–15



Defender	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	1.00

Attacker	aSA-3.0	aVP-0.3-3.0
Equilibrium	0.437	0.563

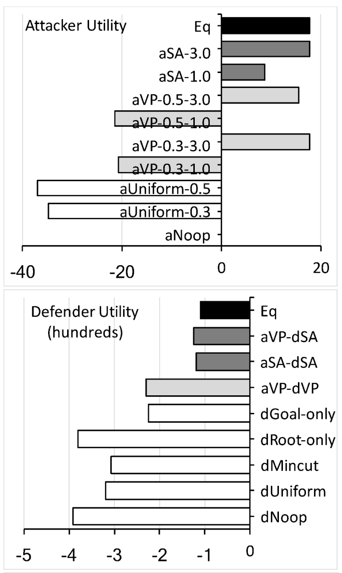
(a) Game 16



Defender	(aSA-1.0)-(dSA-1.0-staG)	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	0.056	0.944

Attacker	aSA-3.0	aVP-0.3-3.0
Equilibrium	0.996	0.004

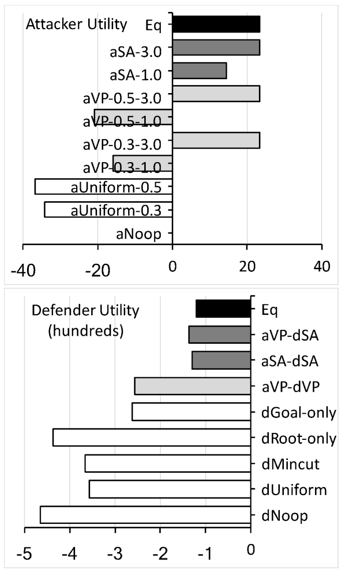
(b) Game 17



Defender	(aSA-1.0)-(dSA-3.0-staG)	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	0.148	0.852

Attacker	aSA-3.0	aVP-0.3-3.0
Equilibrium	0.959	0.041

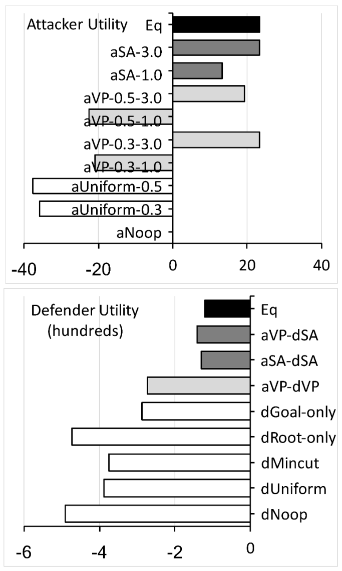
(c) Game 18



Defender	(aSA-1.0)-(dSA-3.0-ranG)	(aVP-0.5-3.0)-(dSA-3.0-ranG)
Equilibrium	0.053	0.947

Attacker	aSA-3.0	aVP-0.5-3.0
Equilibrium	0.993	0.007

(d) Game 19

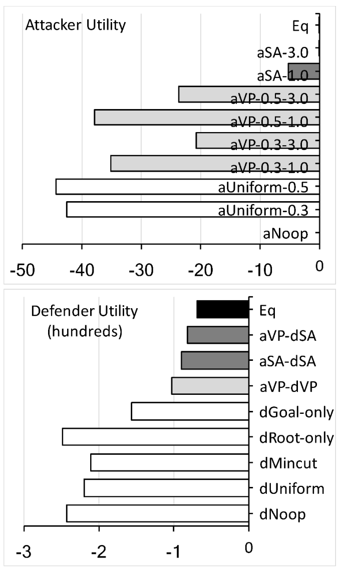


Defender	(aSA-1.0)-(dSA-3.0-ranG)	(aVP-0.5-3.0)-(dSA-3.0-ranG)
Equilibrium	0.005	0.995

Attacker	aSA-3.0	aVP-0.3-3.0
Equilibrium	0.920	0.080

(e) Game 20

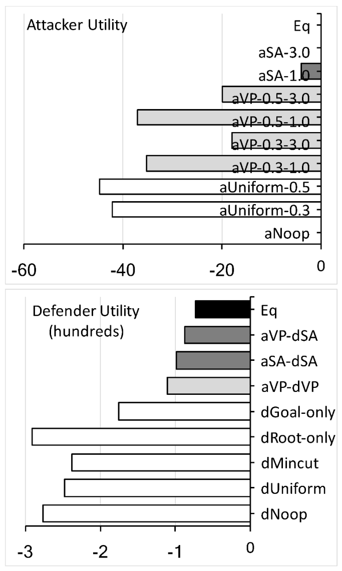
Figure 17: Equilibrium analysis, layered DAGs, 0% $\wedge$-nodes, low noise, games 16–20



Defender	(aSA-3.0)-(dSA-3.0-ranG)	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	0.446	0.554

Attacker	aNOOP	aSA-3.0
Equilibrium	0.394	0.606

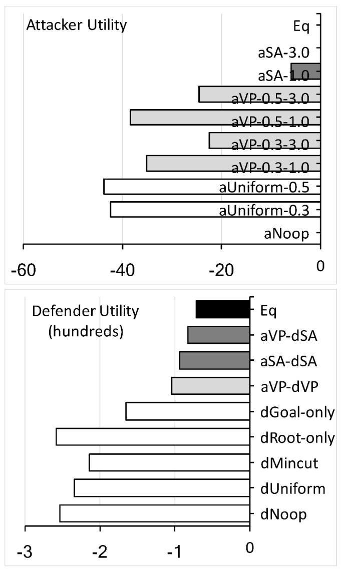
(a) Game 21



Defender	(aSA-3.0)-(dSA-1.0-ranG)	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	0.461	0.539

Attacker	aNOOP	aSA-3.0
Equilibrium	0.327	0.673

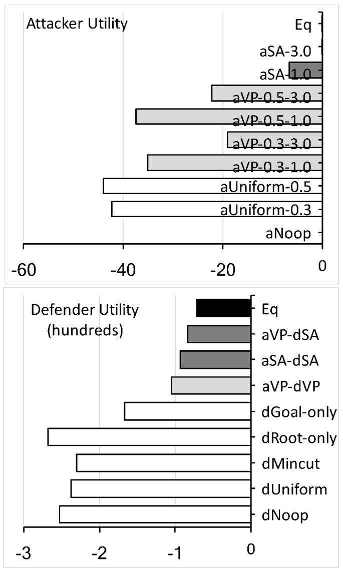
(b) Game 22



Defender	(aSA-3.0)-(dSA-1.0-ranG)	(aVP-0.3-3.0)-(dSA-1.0-ranG)
Equilibrium	0.548	0.452

Attacker	aNOOP	aSA-3.0
Equilibrium	0.385	0.615

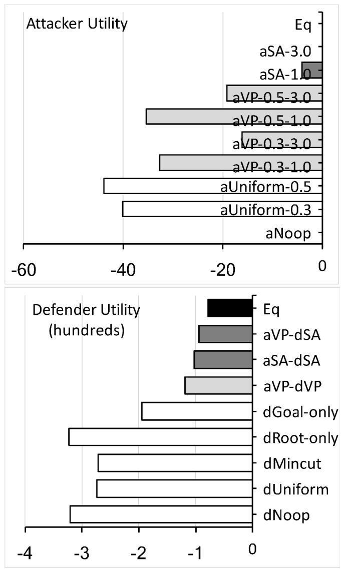
(c) Game 23



Defender	(aSA-3.0)-(dSA-3.0-ranG)	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	0.491	0.509

Attacker	aNOOP	aSA-3.0
Equilibrium	0.415	0.585

(d) Game 24

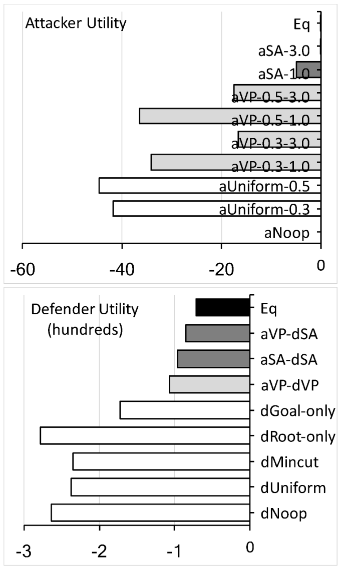


Defender	(aSA-3.0)-(dSA-1.0-ranG)	(aVP-0.5-3.0)-(dSA-1.0-ranG)
Equilibrium	0.439	0.561

Attacker	aNOOP	aSA-3.0
Equilibrium	0.295	0.705

(e) Game 25

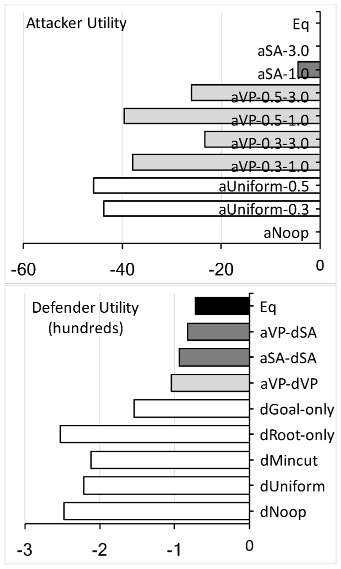
Figure 18: Equilibrium analysis, layered DAGs, 0% $\wedge$-nodes, no noise, games 21–25



Defender	(aSA-3.0)-(dSA-1.0-ranG)	(aVP-0.5-3.0)-(dSA-3.0-ranG)
Equilibrium	0.434	0.566

Attacker	aNOOP	aSA-3.0
Equilibrium	0.360	0.640

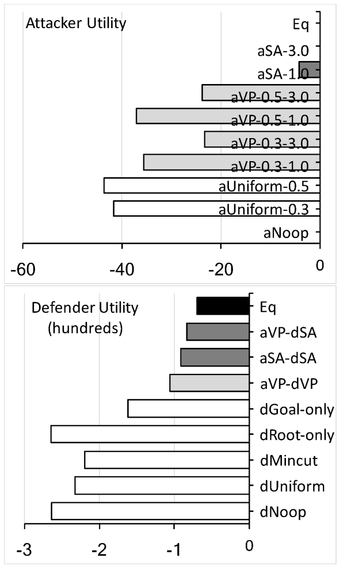
(a) Game 26



Defender	(aSA-3.0)-(dSA-1.0-ranG)	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	0.474	0.526

Attacker	aNOOP	aSA-3.0
Equilibrium	0.457	0.543

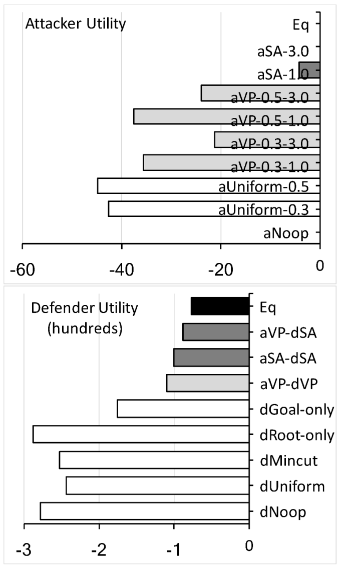
(b) Game 27



Defender	(aSA-3.0)-(dSA-1.0-ranG)	(aVP-0.3-3.0)-(dSA-1.0-ranG)
Equilibrium	0.340	0.660

Attacker	aNOOP	aSA-3.0
Equilibrium	0.331	0.669

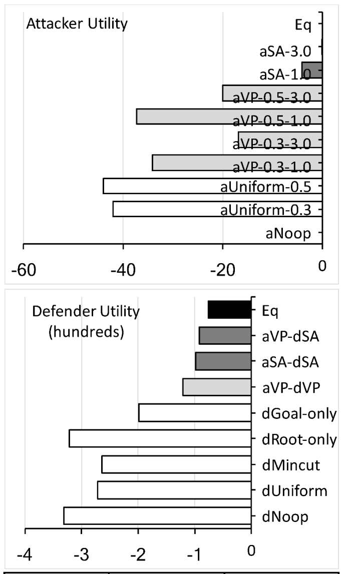
(c) Game 28



Defender	(aSA-3.0)-(dSA-3.0-ranG)	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	0.488	0.512

Attacker	aNOOP	aSA-3.0
Equilibrium	0.394	0.606

(d) Game 29



Defender	(aSA-3.0)-(dSA-3.0-ranG)	(aVP-0.3-3.0)-(dSA-3.0-ranG)
Equilibrium	0.392	0.608

Attacker	aNOOP	aSA-3.0
Equilibrium	0.326	0.674

(e) Game 30

Figure 19: Equilibrium analysis, layered DAGs, 0% $\wedge$-nodes, no noise, games 26–30

Barbara Kordy, Ludovic Piètre-Cambacédès, and Patrick Schweitzer. DAG-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer Science Review*, 13:1–38, 2014.

Igor Kotenko and Andrey Chechulin. A cyber attack modeling and impact assessment framework. In *Cyber Conflict (CyCon), 2013 5th International Conference on*, pages 1–24. IEEE, 2013.

Yu Liu and Hong Man. Network vulnerability assessment using Bayesian networks. In *Defense and Security*, pages 61–71. International Society for Optics and Photonics, 2005.

Kong-wei Lye and Jeannette M Wing. Game strategies in network security. *International Journal of Information Security*, 4(1-2):71–86, 2005.

Richard D. McKelvey, Andrew M. McLennan, and Theodore L. Turocy. Gambit: Software tools for game theory. Technical report, Version 0.2006.01.20, 2006. URL `http://econweb.tamu.edu/gambit/`.

Erik Miehling, Mohammad Rasouli, and Demosthenis 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.

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

Kien C Nguyen, Tansu Alpcan, and Tamer Basar. Security games with incomplete information. In *ICC*, pages 1–6, 2009a.

Kien C Nguyen, Tansu Alpcan, and Tamer Basar. Stochastic games for security in networks with interdependent nodes. In *Game Theory for Networks, 2009. GameNets' 09. International Conference on*, pages 697–703. IEEE, 2009b.

Thanh H. Nguyen, Mason Wright, Michael P. Wellman, and Satinder Baveja. Multi-stage attack graph security games: Heuristic strategies, with empirical game-theoretic analysis. In *Proceedings of the 2017 Workshop on Moving Target Defense*, MTD '17, pages 87–97, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5176-8. doi: 10.1145/3140549.3140562. URL `http://doi.acm.org/10.1145/3140549.3140562`.

Animesh Patcha and J-M Park. A game theoretic approach to modeling intrusion detection in mobile ad hoc networks. In *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*, pages 280–284. IEEE, 2004.

Cynthia Phillips and Laura Painton Swiler. A graph-based system for network-vulnerability analysis. In *Workshop on New Security Paradigms*, pages 71–79. ACM, 1998.

Nayot Poolsapassit and Indrajit Ray. Investigating computer attacks using attack trees. In *IFIP International Conference on Digital Forensics*, pages 331–343. Springer, 2007.

Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray. Dynamic security risk management using Bayesian attack graphs. *IEEE Transactions on Dependable and Secure Computing*, 9(1):61–74, 2012.

Achintya Prakash and Michael P. Wellman. Empirical game-theoretic analysis for moving target defense. In *Second ACM Workshop on Moving Target Defense*, pages 57–65, 2015.

Indrajit Ray and Nayot Poolsapassit. Using attack trees to identify malicious attacks from authorized insiders. In *European Symposium on Research in Computer Security*, pages 231–246. Springer, 2005.

Rudolph van der Merwe, Arnaud Doucet, Nando de Freitas, and Eric Wan. The unscented particle filter. In *Advances in Neural Information Processing Systems*, pages 584–590, 2001.

Stilianos Vidalis, Andy Jones, et al. Using vulnerability trees for decision making in threat assessment. *University of Glamorgan, School of Computing, Tech. Rep. CS-03-2*, 2003.

Lingyu Wang, Anyi Liu, and Sushil Jajodia. Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. *Computer communications*, 29(15):2917–2933, 2006.

Lingyu Wang, Sushil Jajodia, Anoop Singhal, and Steven Noel. k-zero day safety: Measuring the security risk of networks against unknown attacks. In *European Symposium on Research in Computer Security*, pages 573–587. Springer, 2010.

Lingyu Wang, Sushil Jajodia, Anoop Singhal, Pengsu Cheng, and Steven Noel. k-zero day safety: A network security metric for measuring the risk of unknown vulnerabilities. *IEEE Transactions on Dependable and Secure Computing*, 11(1):30–44, 2014.

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

Mason Wright, Sridhar Venkatesan, Massimiliano Albanese, and Michael P. Wellman. Moving target defense against DDoS attacks: An empirical game-theoretic analysis. In *Third ACM Workshop on Moving Target Defense*, 2016.

Xia Zheng You and Zhang Shiyong. A kind of network security behavior model based on game theory. In *Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on*, pages 950–954. IEEE, 2003.

Quanyan Zhu and Tamer 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.

Quanyan Zhu and Stefan Rass. On multi-phase and multi-stage game-theoretic modeling of advanced persistent threats. *IEEE Access*, 6:13958–13971, 2018.

Saman A. Zonouz, Himanshu Khurana, William H. Sanders, and Timothy M. Yardley. RRE: A game-theoretic intrusion response and recovery engine. *IEEE Transactions on Parallel and Distributed Systems*, 25(2):395–406, 2014.