General Sum Markov Games for Strategic Detection of Advanced Persistent Threats using Moving Target Defense in Cloud Networks

Sailik Sengupta, Ankur Chowdhary,
Dijiang Huang, and Subbarao Kambhampati

Arizona State University, Tempe, AZ, USA
{sailiks,achaud16,dijiang,rao}@asu.edu

Abstract. The processing and storage of critical data in large-scale cloud networks necessitate the need for scalable security solutions. It has been shown that deploying all possible detection measures incur a cost on performance by using up valuable computing and networking resources, thereby resulting in Service Level Agreement (SLA) violations promised to the cloud-service users. Thus, there has been a recent interest in developing Moving Target Defense (MTD) mechanisms that helps to optimize the joint objective of maximizing security while ensuring that the impact on performance is minimized. Often, these techniques model the challenge of multi-stage attacks by stealthy adversaries as a single-step attack detection game and use graph connectivity measures as a heuristic to measure performance, thereby (1) losing out on valuable information that is inherently present in multi-stage models designed for large cloud networks, and (2) come up with strategies that have asymmetric impacts on performance, thereby heavily affecting the Quality of Service (QoS) for some cloud users. In this work, we use the attack graph of a cloud network to formulate a general-sum Markov Game and use the Common Vulnerability Scoring System (CVSS) to come up with meaningful utility values in each state of the game. We then show that, for the threat model in which an adversary has knowledge of a defender's strategy, the use of Stackelberg equilibrium can provide an optimal strategy for placement of security resources. In cases where this assumption turns out to be too strong, we show that the Stackelberg equilibrium turns out to be a Nash equilibrium of the general-sum Markov Game. We compare the gains obtained using our method(s) to other baseline techniques used in cloud network security. Finally, we highlight how the method was used in a real-world small-scale cloud system.

1 Introduction

A cloud service provider provides processing and storage hardware along with networking resources to customers for profit. Although a cloud provider might want to use state-of-the-art security protocols, vulnerabilities in software desired (or used) by customers can put sensitive information stored in or communicated over the cloud network at risk. Distributed elements such as firewalls, Intrusion Detection Systems (IDS), log-monitoring systems etc. have been the backbone for detecting malicious traffic in (or stopping them from entering) such systems. Unfortunately, the scale of modern-day cloud systems makes the placement of all possible detecting and monitoring mechanisms an expensive solution [12, 32, 28]; using up the computing and network resources that could have been better utilized by giving them to customers which in turn would be better for business. Thus, the question of how one should place a limited number of detection mechanisms to limit the impact on performance while ensuring that the security of the system is not drastically weakened becomes a significant one.

There has been an effort to answer this question in previous research works [32, 28]. Researchers have pointed out that static placement of detection systems is doomed to be insecure because an attacker, with reconnaissance on their side (which exists for any such cyber-system is by default), will eventually learn this static placement strategy and hence, avoid it. Thus, a dynamic placement of these detection mechanisms, generally known as Moving Target Defense for continuously shifting the detection surface, has become a default. In this method, the set of attacks for which monitoring systems are placed changes in some randomized fashion after every fixed time step. In these works, the cloud system is treated in a way similar to that of physical security systems where the primary challenge is to allocate a limited set of security resources to an asset/schedule that needs to be protected [23, 31, 29].

In the case of real-world cloud-systems, these aforementioned solutions lead to three problems. First, only single-step attacks are considered in the game-theoretic modeling which lead to sub-optimal strategies because such models fail to account for multi-stage attack behavior. For example, strategies generated by prior work may prioritize detecting a high-impact attack on a web-server more than a low-impact attack on a path that leads an attack on a storage server, which when exploited may have major consequences. Second, the threat model assumes that an attacker can launch an attack from any node in the cloud network. This is too strong an assumption, leading to sub-optimal placement strategies in real-world settings. Third, existing methods can come up with placement strategies that allocate multiple detection systems on a sub-net while another sub-net is not monitored. This results in a steep degradation of performance for some customers. To address these challenges, we need a suitable method for modeling multi-stage attacks. Unfortunately, capturing all possible attack paths can lead to an action-set explosion in previously proposed normal-form games [28]. Thus, we use *Markov Games* to model such interactions.

Specifically, we try to address these problems by modeling the cloud system as a General-Sum Markov Game. We use particular nodes of our system's attack graph to represent the states of our game while the attacker's actions are modeled after real-world attacks based on the Common Vulnerabilities and Exploits (CVEs) described in the National Vulnerability Database (NVD) [1]. The defender's actions correspond to the placement of detection systems that can detect the attacks. We design the utility values for each player in this game leveraging (1) the Common Vulnerability Scoring Systems, which provide metrics for each attack, and (2) cloud designer's quantification of how the placement of a detection system impacts the performance. These help us come up with defense strategies that take into account the long-term impacts of a multi-stage attack while restricting the defender to pick a limited number of monitoring actions in each part of the cloud. The latter constraints ensure that the performance impact on a customer, due to the placement of detection measures, is limited.

The popular notion of using min-max equilibrium for Markov Games [18] is an optimal strategy for the players in zero-sum games and becomes sub-optimal for a general-sum game. Furthermore, we model an attacker who is aware of the defender's placement strategy at each state of our Markov game and thus, consider the Stackelberg equilibrium of this game. In scenarios where the latter assumption is too strong, we show that the Stackelberg Equilibrium of our general-sum game is, depending on the problem structure, a subset of Nash Equilibria and still results in optimal strategies. The key contributions of this research work are as follows:

- We model the multi-stage attack scenarios, which are typically employed in launching Advanced Persistent Threats (APTs) campaigns against high-value targets, as a general-sum Markov Game. The cost-benefit analysis based on the two-player game, provides strategies for placing detection systems in a cloud network.
- We leverage the attack-graph modeling of cloud networks, the Common Vulnerabilities and Exposures (CVEs) present in the National Vulnerability Database and the Common Vulnerability Scoring Service (CVSS) to design the states, the actions and utility values of our game. In addition, we consider prior work in cloud-systems to (1) model the uncertainty of an attack's success and (2) leverage heuristic measures that model the performance impact of placing detection mechanisms on the cloud.
- Our framework considers a threat model where the attacker can infer the defender's detection strategy. Therefore, we design a dynamic programming solution to find the Stackelberg equilibrium of the Markov Game. If an attacker does not have information about the defender's strategy, we show that the Stackelberg equilibrium of the general-sum Markov Game is a subset of Nash Equilibrium when a set of properties hold (similar to prior work in extensive form games [16]). In order to showcase the effectiveness of our approach we analyze a synthetic and a real-world cloud system.

2 Background

In this section, we first introduce the reader to the notion of real-world vulnerabilities and exploits present in a cloud system that we will use throughout our paper. Second, we describe the threat model for our cloud scenario. Lastly, we describe the notion of Attack Graphs (AG) followed by a brief description of Markov games and some well-known algorithms used to find the optimal policy

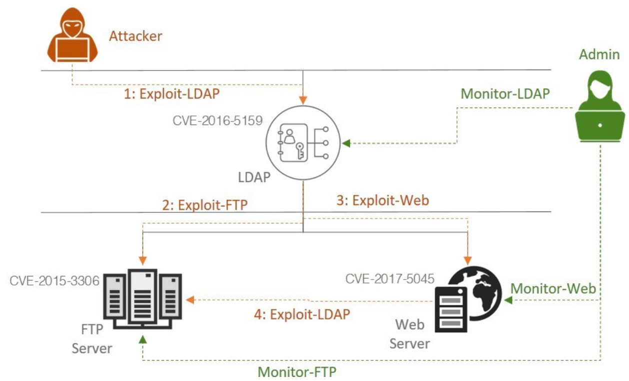

Fig. 1. An example cloud system highlighting its network structure, the attacker and defender (admin) agents and the possible attacks and monitoring mechanisms.

or strategy for each player. We will use the example attack scenario for cloud networks shown in Figure 1 as a running example in our discussion.

2.1 Vulnerabilities and Exploits

Software security is defined in terms of three characteristics - Confidentiality, Integration, and Availability [20]. Thus, in a broad sense, a vulnerability (that can be attacked or exploited) for a cloud system can be defined as a security flaw in a software service hosted over a given port. When exploited by a malicious attacker, it can cause loss of Confidentiality, Availability or Integrity (CIA) of that virtual machine (VM). The National Vulnerability Database (NVD) is a public directory of known vulnerabilities and exploits. It assigns each known attack a unique identifier (CVE-id), describes the technology affected and the attack behavior. Thus, to model the known attacks against our system, we use the Common Vulnerabilities and Exposures (CVEs) listed in NVD.

In the cloud-scenario described in Figure 1, we have three VMs– an LDAP server, an FTP server, and a Web server. Each of these servers have a (set of) vulnerability present on it. On the LDAP server, an attacker can use a local privilege escalation to gain root privilege on it. The other two vulnerabilities– A cross-side scripting (XSS) attack on the Web server and the remove code execution on the FTP server– can only be executed with root access to the LDAP server. We can now describe the threat model for our scenario.

2.2 Threat Model

In the example cloud scenario, the attacker starts with user-level access to an LDAP server. The terminal state is to compromise the FTP server (which, as we will see later, leads to an all absorbing state in our Markov Game). The attacker can perform actions such as `1: exploit-LDAP`, `exploit-Web` or `exploit-FTP`. Note that the attacker has two possible paths to reach the goal node, i.e. `priv(attacker, (FTP: root))` which are:

– Path 1: `exploit-LDAP` $\rightarrow$ `exploit-FTP`
– Path 2: `exploit-LDAP` $\rightarrow$ `exploit-Web` $\rightarrow$ `exploit-FTP`

On the other hand, the (network) Admin, who is the defender in our case, can choose to monitor (1) read-write requests made by services running on a VM using host-based Intrusion Detection Systems (IDS) like `auditd`, or (2) network traffic along both the paths using the network-based monitoring agents like `Snort`. We will denote these IDS systems using the terminology `monitor-LDAP`, `monitor-FTP`, etc. We assume that the Admin has a limited budget, i.e., cannot place all possible IDS system on the cloud network, and thus, must try to perform monitoring in an optimized fashion. On the other hand, the attacker will try to perform attacks along with some path that minimizes their probability of getting detected. Further, we assume an attacker has knowledge of the defender's placement strategy because of the inherent reconnaissance phase in cyber-security scenarios, thus rendering pure strategies for placement of detection systems useless. Thus, to come up with a good dynamic placement placement strategy, we need to model the various multi-attack paths and the attacker's strategy. We first discuss the formalism of Attack Graphs that are a popular way to model the various attacks (and attack paths) in a cloud scenario [12, 4] and then give a brief overview of two-player Markov Games.

2.3 Attack Graph Formalism

Attack Graphs (AG) are a representation tool used to model the security scenario of a complex network system like the cloud. Researchers have shown that AG can help to model multi-stage or multi-hop attack behavior.
Attack Graph is a graph $G = \{N, \mathcal{E}\}$ that consists of a set of nodes (N) and a set of edges ($\mathcal{E}$) where,

– As shown in the Figure 2, nodes can be of four types– the nodes N_C represent vulnerabilities (shown as rectangles), e.g. `vulExists (LDAP, Local Priv. Escalation)`, N_D represents the attacker's state (shown as diamonds) e.g., `priv(attacker, (LDAP :user))`, rule nodes N_R represent a particular exploit action (shown as ellipses) and finally, root or goal nodes N_G that represent the goal of an attack scenario (shown using two concentric diamonds), e.g., `priv(attacker, (FTP: root))`.
– $\mathcal{E} = \mathcal{E}_{pre} \times \mathcal{E}_{post}$ denotes a set of directed edges in the graph. An edge $e \in \mathcal{E}_{pre}$ goes from a node in N_D or N_C to a rule node N_R and denotes that an attack or rule can only be triggered if all the conditions of the edges going into $n \in N_R$ is satisfied (AND-nodes). An edge $e \in \mathcal{E}_{post}$ goes from a node N_R to $n \in N_D$ indicating the change in attacker's privilege changes upon successful execution of the rule.

Note how the two attacks paths mentioned in the threat model section become evident by a simple look at the AG. The conditional and cumulative probability values pertaining to the success of an attack path over the AND (conjunct) and OR (disjunct) nodes can be calculated using probability estimates as described by Chung *et. al.* [6].

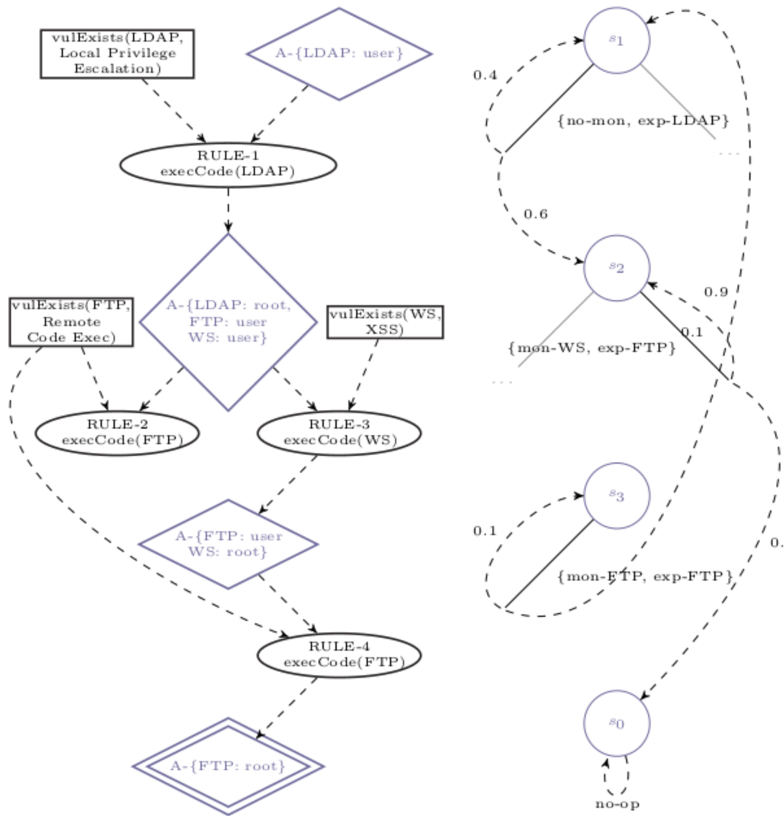

Fig. 2. The left figure shows the attack graph of the synthetic cloud scenario shown in 1. The right figure shows the formulated Markov Game.

2.4 Two-Player Markov Games

We now define a two-player Markov Game and also introduce the reader to some notations used later in our modeling. We call the two players of this game the Defender D (who is the Admin of the cloud system) and the Attacker A (who is an adversary trying to exploit a vulnerability in the Cloud System). With that, we can now formally define a Markov Game as follows.
Markov Game for two players D and A can be defined by the tuple $(S, M, E, \tau, U^D, U^A, \gamma^D, \gamma^A)$ [30] where,

- $S = \{s_1, s_2, s_3, \ldots, s_k\}$ are finite states of the game,
- $M = \{m_1, m_2, \ldots, m_n\}$ is the finite set of monitoring actions for D,
- $E = \{e_1, e_2, \ldots, e_{n'}\}$ is the finite set of (exploit) actions available to A,
- $\tau(s, m_i, e_j, s')$ represents the probability of reaching a state $s' \in S$ from the state $s \in S$ if D chooses to deploy the monitoring m_i and A chooses to use the exploit e_j,
- $U^i(s, m_i, e_j)$ represents the reward obtained by player $i(= A$ or $D)$ if in state s, D choose to deploy the monitoring m_i and A choose to use the exploit e_j,

VM	Vulnerability	CVE	CIA Impact	Attack Complexity
LDAP	Local Priv Esc	CVE-2016-5195	5.0	MEDIUM
Web Server (WS)	Cross Site Scripting	CVE-2017-5095	7.0	EASY
FTP	Remote Code Exec.	CVE-2015-3306	10.0	MEDIUM

Table 1. Vulnerability Information for the Cloud Network

– $\gamma^i \mapsto [0, 1)$ is the discount factor for player $i(= A$ or $D)$.

In light of recent studies on characterizing attackers based on personality traits [2], one might argue that a defender's perspective of long term rewards is different than that of an attacker. Given that we did not find a formal model or user study clearly stating how these differ, we will consider $\gamma^A = \gamma^D = \gamma$ going forward. As the solvers for our formulated game can work in cases even when $\gamma^A \neq \gamma^D$, this assumption just helps us simplify the notations.

The concept of an optimal policy in this Markov game is well-defined for zero-sum games [18] where $U^D(s, m_i, e_j) = -U^A(s, m_i, e_j) \ \ \forall\ s \in S, m_i \in M$, and $e_j \in E$. In these cases, a small modification to the Value Iteration algorithm can be used to compute the min-max strategy for both players. To see this, note that the Q-value update for this Markov Game (for player x) becomes as follows,

$$Q^x(s, m_i, e_j) = R^x(s, m_i, e_j) + \gamma \sum_{s'} \tau(s, m_i, e_j, s') \cdot V^D(s') \tag{1}$$

where $V^D(s')$ denotes the value function (or reward-to-go) with respect to D if in state s'. We will use the notation $M(s)$ (and $E(s)$) to denote the set of defender actions (and attacker actions) possible in state s. Given this, the mixed policy $\pi(s)$ for state s over the defender's applicable actions ($\in M(s)$) can be computed using the value-update,

$$V^x(s) = \max_{\pi(s)} \min_{e_j} \sum_{m_i} Q^x(s, m_i, e_j) \cdot \pi_{m_i} \tag{2}$$

where π_{m_i} denotes the probability of choosing the monitoring strategy m_i.

When the Markov Game has a general-sum reward structure and one player can infer the other player's strategy before making their move, the min-max strategy becomes sub-optimal and one must consider other notions of equilibrium [33, 9]. We give an overview of prior work on these lines later in the paper.

2.5 Quantifying the Impact of Vulnerabilities

The use of the Common Vulnerability Scoring System (CVSS) for rating the impact of attacks is well studied in cyber-security [10, 29]. For (almost all) CVEs listed in the NVD database, we have a six-dimensional CVSS v2 vector,which can be decomposed into multiple measures like Access Complexity (AC) that models the difficulty of exploiting a particular vulnerability and the impact on Confidentiality, Integrity, and Availability (CIA score) gained by exploiting it.

The values of AC are categorical {`EASY, MEDIUM, HIGH`} (that have a corresponding numerical value associated with them), while CIA values are in the range $[0, 10]$. For the set of vulnerabilities present in our system, the values of the two metrics are shown in Table 1.

3 Markov Game Modeling

Before discussing the game theoretic formulation in detail, we highlight a few important assumptions. Besides the Markovian assumption, we assume that (1) there is a list of attacks known to both the attacker and the defender (which cannot be immediately fixed either due to lack of resources or restrictions imposed by third-party customers who host their code on the cloud system [11, 28]) and (2) the attacker may reside in the system but will remain undetected until it attempts to exploit an existing vulnerability, i.e. a *stealthy adversary* [32]. These assumptions forces our formulation to (1) only deal with known attacks and, (2) come up with good (but in-optimal) strategies for placing detection mechanisms. The latter is a result of ignoring the partial observability inherent in the problem [21, 22] to come up with scalable solutions, necessary for cloud networks.

States The state of our Markov Game (MG) are derived using the nodes N_D and N_G of an Attack Graph (AG) (the blue diamond shaped nodes are mapped to the blue circular nodes in Fig. 2). These nodes in the AG represent the state of an attacker in the cloud system, i.e. the attacker's position and their privilege level. The goal or a root node N_G of an AG are mapped to a terminal self-absorbing state in our MG while the other nodes N_D represent non-terminal game states. Note that the location of an attacker on multiple physical servers in the cloud network can map to a single state of the AG (and therefore, a single state in our MG). The MG states that map to a goal or a root node N_G of an AG are the terminal self-absorbing states. Among these non-terminal states there exist a set of states S_i that represent the external-facing entry-points to the cloud network, the initial state of any multi-stage attack.

For the cloud scenario shown in Fig 1, we have four states. The state s_0 corresponds to the goal node `A-{FTP:root}` and is a terminal state of the MG. The state $s_1 (\subset S_i)$ corresponds to the state where an attack originates while the two states s_2 and s_3 correspond to nodes where the attacker has different access privileges on the three servers (LDAP, WS and FTP) server. Given that we use a ternary range to denote an adversary's privilege– no-access, user or root-user– on each server, there can be a maximum of nine states (# servers $\times$ # access-levels)[1] in the AG and hence, in our MG. Note that given a set of known attacks, the number of states is often much less (four *vs.* nine for our scenario) because most of the states are not reachable from the states in S_i.

[1] Partial observability over the state space can increase the number of states to be a power-set of this number, i.e. $2^{(\#\ \text{servers} \times \#\ \text{access-levels})}$

Players and Pure Strategies As mentioned above, the players for our game are the Admin (or the defender) D and the attacker A. The pure strategy set for A in state s consists of exploit actions they can perform with the privilege they have in state s. For example, consider s_2 where the he attacker has the access vector `A-{LDAP:root, FTP:user, WS:user}`. With this as the precondition, the attack actions can be represented by the rule nodes N_R (shown in oval) in the AG. Note that there is always a vulnerability node $\in N_C$ associated with a rule node and thus, with each action.

The pure strategy set for D in a state s consists of monitoring actions where each such action corresponds to placing an Intrusion Detection System (IDS) for detecting attacks that can be executed by A in state s. These actions are made possible in real-world scenarios by using two sorts of IDS systems– (1) host based IDS like *auditd* that can notify D about abnormality in the use of CPU resources or access to files on the server and (2) network based IDS like *snort* that can observe traffic on the wire and report the use of unexpected bit patterns in the header or body of a packet. Although a pure strategy for D can only detect a single attack in our simple example, it is possible that a set of detection systems, capable of detecting multiple attacks, is considered as a pure strategy. We will see this in the case of the real-world cloud scenario discussed in the experimental section. In the context of Stackelberg Security Games, such groups of actions are often called schedules [15, 24] and the pure strategy is defined over these schedules. We note that our modeling supports such representations.[2]

To allow for a realistic setting, we add one more action to the pure strategy set of each player–`no-act` and `no-mon`. These represent the no-op for each player which allows an attacker to not attack in a particular state if it feels that there is a high risk of getting caught. Similarly, this allows a defender to not monitor for an attack thereby saving valuable resources.

Transitions The transitions in our MG represent that given a particular state and a pair of actions drawn from the joint action space $E \times M$, the probability with which a game reaches a state s', i.e. $\tau(s, m, e, s')$. There exists a few obvious constraints in the context of our MG– (1) $\tau(s, m, \texttt{no-act}, s) = 1$, i.e. if an attacker does not execute an attack, the game remains in the same state, (2) when $e \neq \texttt{no-act}$, $\tau(s, m_e, e, s') = p/|S_i| \; \forall s' \in S_i$ where p is the probability that e is detected by m_e, the monitoring service deployed to detect e, i.e. when successfully detected, the attacker starts from either of the initial states with equal probability, and (3) $\tau(s, \texttt{no-mon}, e, s') \neq 0$ if $s \notin S_i$ and $s' \in S_i$, i.e. an attacker cannot be detected if the defender does not perform a monitoring action.

We highlight a few transitions for our Markov Game in Fig 2. In state s_1, the defender does not monitor the exp-LDAP attack action and with 0.6 probability the game moves to the state s_2 (and with 0.4 it remains in s_1). These probability are calculated using the Access Complexity vector and the function

[2] In these cases, the Subset of Sets are Sets (SSAS) property defined in [16] may not hold and thus, the Strong Stackelberg Equilibrium will not always be a Nash Equilibrium for the formulated Markov Game (see later (*Lemma 1*) for details).

		D (Defender)		
		no-mon	mon-Web	mon-FTP
	no-act	$0, 0$	$0, -2$	$0, -3$
A (Attacker)	exp-Web	$7, -7$	$-8, 6$	$7, -10$
	exp-FTP	$10, -10$	$10, -12$	$-8, 5$

Table 2. Utilities (U^A, U^D) for state s_2

	no-mon	mon-LDAP
no-act	$0, 0$	$0, -2$
exp-LDAP	$5, -5$	$-5, 3$

	no-mon	mon-FTP
no-act	$0, 0$	$0, -2$
exp-FTP	$10, -10$	$-8, 6$

Table 3. Utilities (U^A, U^D) for states s_1 (left) and s_3 (right).

defined in [6] for obtaining the probability of success given an attack. This is also done when the defender deploys a monitoring mechanism m_e but the attacker executes another attack e' where $e \neq e'$ and $e' \neq$ `no-act` (see the transition for an example joint action from s_2 in Fig 2). Lastly, the transition from s_3 shows a case relating to (3) mentioned in the previous paragraph. The value of $\tau(s_3, \text{mon-FTP}, \text{exp-FTP}, s_1) = 0.9$ because monitoring access to files like `/etc/passwd` and `/etc/shadow` can only detect some forms of privilege escalations (such as remote code execution that tries to eithers creates a new user or tries to escalate the privilege of a non-root user), but may not be able to monitor a case where an attacker simply obtains access to a root user account.

Rewards The rewards that can be obtained by the players depending on their strategy in a particular state (except the terminal state s_0) of our cloud scenario is shown in Table 3 and 2. Most prior works [26, 22, 32] haven't used sensible heuristics to come up with attacker utility and or defender's resource costs. In our case, the reward values are obtained using multiple metrics– (1) the impact score (IS) (also called the CIA impact score) of a particular attack, (2) the cost of performance degradation (provided by security and engineering experts in the cloud domain; often obtained by running MiniNET simulation) based on the placement of a particular IDS [28], and (3) the hops taken by an attacker to reach a particular state, which is often used to measure how advanced an Advanced Persistent Threat (APT) is. Note that the last factor is non-Markovian part in the overall reward function of our Markov Game; it depends on the path taken by an attacker to reach a particular state. To bypass this issue, we consider all possible paths an attacker can take to reach the particular state and average the path value, which gives us an average of how advanced is the APT. Further, the actual path taken by a stealthy adversary who has been residing in the network for a long time, is difficult (if not impossible) to obtain. Thus, an average estimate is a good heuristic for estimating the importance of an APT.

We will not explain how the reward value for the action pair `(mon-Web, exp-Web)`, shown in Table 2, was obtained. First, The impact score for this vulnerability CVE-2017-5059, shown in Table 1, is 7. Second, we monitored performance using `Nagios` [8] to measure the end-to-end network bandwidth, number

Algorithm 1 Dynamic Programming for finding SSE in Markov Games

1: **procedure** GIVEN $(S, M, E, \tau, U^D, U^A, \gamma^D = \gamma^A = \gamma)$,
2: OUTPUT$(V^i(s), \pi^i(s) \ \forall \ i \in \{A, D\})$
3: $\quad V(s) = 0 \ \forall \ s$
4: $\quad$ *loop*: $i == k$ break;
5: $\quad$ // Update Q-values
6: $\quad$ Update $Q^D(s, m, e)$ and $Q^A(s, m, e) \ \ \forall \ s \in S, m \in M(s), e \in E(s)$
7: $\qquad$ using U^D, U^A and $V(s)$.
8: $\quad$ // Do value and policy computation
9: $\quad$ Calculate $V^i(s)$ and $\pi^i(s)$ for $i \in \{A, D\}$ using the values $Q^i(s, m, e)$ in Eq. 3
10: $\quad i \leftarrow i + 1$
11: $\quad$ **goto** *loop*.
12: **end procedure**

of concurrent requests to live web-services and the delay in servicing network requests when mon-Web was deployed. We observed that there was an increase in network delay, decrease in network bandwidth and decrease in the number of concurrent requests serviced per second. Based on expert knowledge, we estimated the reward (or rather impact on performance) of placing the IDS that monitors this vulnerability is -2. Finally, given that this vulnerability can only be executed if the attacker has exploited at least one vulnerability before coming to this state, the APT score was calculated to be 1. Thus, the defender's reward's for placing the correct IDS that can detect the corresponding attacker action is 7 minus 2 (cost incurred due to reduced performance) plus 1 (for detecting an APT that had already penetrated 1-hop into the network), totaling 6. On the other hand, the attacker's reward for this action pair is -7 spending effort is executing a vulnerability of impact 7 plus -1 for losing a vantage point in the cloud network, totaling a reward of -8. The other reward values were defined using a similar line of reasoning. Given that the defender's cost of placing IDS is not of any concern for the attacker[3], when an attacker chooses `no-act`, A's reward is 0. On the contrary, the defender will still incur a negative reward if it deploys a monitoring system because it impacts the performance of the sub-net.

3.1 Optimal Placement Strategy

Finding the optimal solution to a two-player general-sum Markov Games is more involved than finding the min-max strategy (in zero-sum settings). Moreover, in our threat model, we assume that the attacker A, with reconnaissance efforts, will get to know the strategy of the defender D, imparting it the flavor of leader-follower games.

We highlight a dynamic programming approach shown in Algorithm 1. Although this algorithm looks similar to the one used for computing min-max equilibrium, it has an important difference. In line 9, instead of using equation 2 to calculate the optimal value and player strategies, we compute the Strong

[3] This is a strong reason to move away from the zero-sum reward modeling in [5].

Stackelberg Equilibrium (SSE) in each state. For each iteration, we first consider the Q-values for that state and the joint actions represented as a normal-form game matrix. We then find the optimal policy for both players in state s. Since our model, at present, does not consider multiple adversary types, the equilibrium calculation for each state can be done in polynomial time [15]. This type of a solution resembles the idea of finding Stackelberg Equilibrium in discounted stochastic games which has been discussed in [33]. They authors describe an ILP approach over all the states of the Markov Game that becomes computationally intensive in our case given the large number of states in our formulation. Furthermore, the iterative approach provides an anytime solution which can be stopped at a premature stage (by setting lower values of k in Algo. 1) to yield a strategy for placement. For completeness, we briefly highlight the optimization problem in [23] that we used to update the value of each state in Algo. 1.

$$\max_{\pi^D,\pi^A} \sum_{m\in M}\sum_{e\in E} Q^D(s,m,e)\pi^D_m\pi^A_e \tag{3}$$

$$s.t. \quad \sum_{m\in M} \pi^D_m = 1,\ \forall \pi^D_m\ \pi^D_m \in [0,1] \qquad \text{Defender's selects a valid mixed strategy.}$$

$$\sum_{e\in E} \pi^A_e = 1,\ \forall \pi^A_e\ \pi^A_e \in \{0,1\} \qquad \text{Attacker's selects a valid pure strategy.}$$

$$0 \leq v - \sum_{m\in M} Q^A(s,m,e)\pi^D_m \leq (1-\pi^A_e)L \quad \forall \pi^A_e \qquad \text{Attacker's pure strategy maximizes their reward given defender's mixed strategy.}$$

where L is a large positive number.

The assumption that an attacker, with the inherent advantage of reconnaissance, is aware of the defenders mixed policy in each state can be a strong one in the context of Markov games. Thus, one might question the optimality of the strategy that we come up with using Algorithm 1. In [17], researchers have shown that SSEs are a subset of Nash Equilibrium for a particular class of problems. Specifically, they show that if the security resource allocation problem (which in our case, is allocating IDS for covering a vulnerability) has a particular property, termed as SSAS, then the defender's SSE is also a NE for the game. Given this, we can state the following.[4]

Lemma 1. *If the Subset of Set Are Sets (SSAS) property holds in every state s of a Markov Game, then the SSE is also a NE of the Markov Game.*

Proof. We will prove the lemma by contradiction. Let us assume that SSAS property holds in every state of a Markov Game (MG), but the SSE of the MG is not the NE. First, consider $\gamma = 0$ for this MG. Thus, the SSE and NE strategy for each state can be calculated only based on the utilities of only this state. Now, if SSE $\notin$ NE for this Markov Game, then there is some state in which the SSE

[4] In the case of multiple attackers, SSE $\notin$ NE. Although such scenarios exist in cybersecurity settings, we consider a single attacker in this modeling and plan to consider the multiple attacker setting in the future.

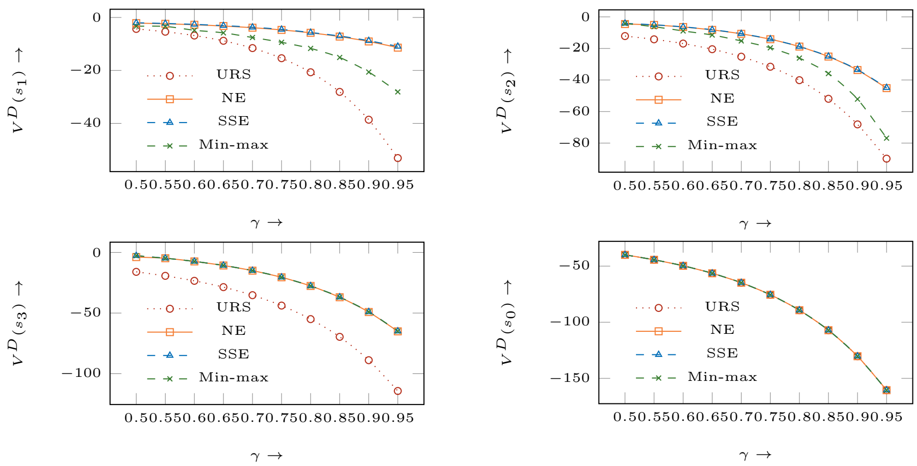

Fig. 3. Defender's value for each of the four state–s_1 (top-left), s_2 (top-right), s_3 (bottom-left), and s_0, which in the all absorbing terminal state (bottom-right).

strategy is not the NE strategy. But if that is the case, then we would have violated the SSAS theorem in [17] for state, which cannot be true. For the case $\gamma > 0$, the proof still holds because note that the SSAS property is not related to the Q-values using which the strategy is computed in the Markov Game. □

Note that this holds trivially for our small example because the defender's pure strategy for each state is to deploy a single IDS (and thus, all subset of schedules are are a possible schedule).

4 Experimental Results

In this section, we first compare the effectiveness of the optimal strategies for our general-sum leader-follower Markov-game against the Uniform Random Strategy, which is a popular method and often used as a baseline for Moving Target Defense strategies in cybersecurity. We then discuss the set-up of a real-world, small scale cloud scenario and the gains we can obtain using our formulated Markov Game.

4.1 Evaluation of Strategies

We first discuss two baseline strategies and then briefly explain why we choose it for comparison with the SSE strategy for our general-sum leader-follower markov-game formulated in the previous section.

• *Uniform Random Strategy (URS)* In this, the defender samples an action by drawing from a uniform random distribution over pure-strategies. For example, in the state s_1 shown in Table 2, the defender can choose to monitor the FTP or the Web server or neither of them, all with an equal probability of 0.33.

Researchers have claimed that selecting between what to choose when shifting between MTD configurations should be done using a uniform random strategy [34]. Although there have been other methods based on Stackelberg Security Games (SSGs) which have shown that such a strategy may be sub-optimal [28], it provides a baseline strategy that can be used in the context of our Markov Game. Adapting the latter strategies proposed in previous work need us to compile our multi-stage Markov into a single step normal form game. First, there is no trivial way of doing this conversion as the rewards in [28] talk about average impact on the network that are difficult to encode meaningfully in our Markov Game. Furthermore, the pure strategy set in [28] would have to incorporate full attack paths as opposed to single attack actions. This would make the strategy computation time-consuming. Second, our work can be seen as a generalization of applying the notion of Stackelberg Equilibria, similar to [33], for Markov Games in the context of IDS placement and thus, a counterpart solution to the normal form games case described in [28] to Markov Games. Hence, we do not consider [28] as a baseline.
• *Min-max Strategy* Although our game is a general sum setting, one might ask how sub-optimal the min-max strategies for a similar zero-sum Markov game is when we ignore the impact on performance. In essence, the attacker still has the same utility in the individual states shows in Tables 3 and 2, but the defender's reward values are just the opposite of the attacker's reward, making it a zero-sum game. Here, we hope to see that the impact on performance would reduce the defender's overall utility.

Comparison of Strategies In Figure 3, we plot the values of the four states ($V(s)$) of our game for the baselines (URS and Min-max), and the Strong Stackelberg Eq. (SSE). We also, to provide empirical support for Lemma 1, plot the Nash Eq (NE). On the x-axis, we vary the discount factor and on the y-axis plot the value of the state with respect to the defender. In the terminal state s_0, the defender gains a high negative reward because the attacker was able to exploit all the possible vulnerabilities successfully without getting detected. Thus, for all the states, since there is non-zero probability of reaching s_0, the defender's value function is negative. Note that as one weighs the future rewards more, i.e. γ approaches 1, the value of the states decrease in magnitude because the negative reward in s_0 is given higher weight.

As stated above, the SSE for our example game is the same as the NE for our Markov Game and the curves for both the strategies overlap. On the other hand, URS is much worse off than our strategy for all the states with more than a single action whereas, Min-max, although better than URS, is sub-optimal with respect to SSE for all states except s_3 and s_0. s_0, being a terminal state, has only one action for the defender and thus, all the methods are trivially equivalent. Thus, all the curves overlap (bottom-right in Fig 3). In state s_3, which is just a single action away from the terminal state with high negative reward, the defender always picks the action to monitor for an attack regardless of the performance impact (whose magnitude is less in comparison to the impact of the attack).

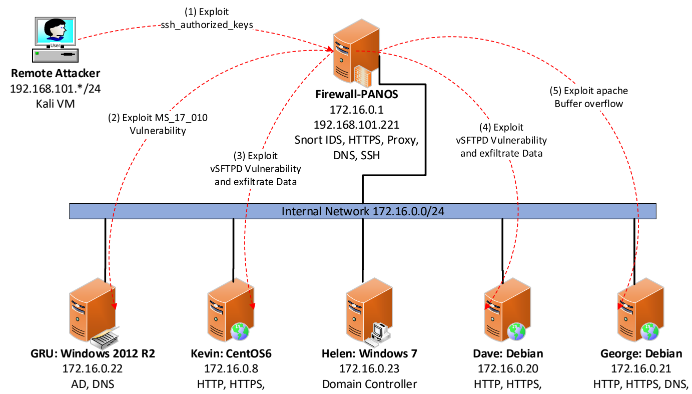

Fig. 4. A real world cloud system.

Thus, even though the Min-max strategy is ignorant of the latter costs, it picks the same strategy as SSE. Hence, their plots overlap (bottom-left in Figure 3). Now, before discussing the differences between SSE and Min-max in the other two states (s_1 and s_2), we take a look at the mixed strategy obtained by finding the SSE of our game. This will help us in explaining the sub-optimality of the Min-max strategy. For a discount factor of $\gamma = 0.8$:

$\pi_{\text{MG-SSE}}(s_0)$: {terminate: 1.0}
$\pi_{\text{MG-SSE}}(s_1)$: {no–mon: 0.097, mon–LDAP: 0.903}
$\pi_{\text{MG-SSE}}(s_2)$: {no–mon: 0.0, mon–Web: 0.539, mon–FTP: 0.461}
$\pi_{\text{MG-SSE}}(s_3)$: {pi_no–mon: 0.0, mon–FTP: 1.0}

Note that, in our example, barring the terminal state s_0, other states have only one or two proper detection actions because `no-mon` asks the defender to not monitor for any attacks. Thus, we expected that these actions to have probabilities almost equal to zero (unless it has a considerable impact on performance). In the case of state s_1, the 0.097 probability of picking that action shows that in states far away from the terminal, the defender chooses to be a little more relaxed in terms of security and pays more attention to performance. On the contrary, in state s_3 an exploit action will move the game to the terminal state that has high-negative reward for the defender and thus, the defender is asked to place all attention to security. In general, this implies that an Admin D should pay more attention to security against APTs deep within the network in states closer to a critical resource and can choose to reason about performance near the entry points of the cloud system. Thus, in these states, the optimal mix-max strategy, oblivious to the performance costs, invests in monitoring resources and thus, becomes sub-optimal with respect to the SSE.

4.2 Case Study on the Cloud

In this section, we do a case study on a real-world sub-network in a cloud system, highlighting briefly the system setup, the Markov Game formulation, the comparison between URS and SSE for a (cherry-picked) state and how all these strategies can be implemented with the help of Software Defined Networking.

Implementation details. We utilized the Virtual Machine (VM) images from the Western Region Cybersecurity Defense Competition (WRCCDC) [7]. The competition helps university students (the Blue Team) gain first-hand experience in dealing with real-world attacks. Students are asked to defend a corporate infrastructure against experienced white-hat hackers (the Red-Team). In the scenario shown in Figure 4, a Red Team attacker mounts a multi-step attack by following a slow and low approach, that tries to evade the IDS placed.

The goal of the attacker can be either to disrupt the network services or ex-filtrate data out of the private networks. Both of these attacks, if successful, can lead to the loss of mission-critical information (FTP server files are valuable to the company) and business (service downtime). We model each of these goal states in the attack graph as states that lead to an all-absorbing terminal state with unit probability, thus ending the game.

The set of attacks were discovered using low-intensity network scanning tools like OpenVAS that run over an extended period of time and generate a report of the vulnerabilities present in the system. Due to space considerations, we summarize the vulnerabilities present in our cloud network in Table 5. Corresponding to the attacks, we considered the deployment of IDS mechanisms like Snort IDS, Web Proxy, etc. situated at different levels of the protocol stack. We used WRCCDC's VM images to create a similar environment in our organization's cloud service. To connect these VMs, we created a flat structure network with *Palo Alto Network OS (Next-Generation Firewall)* hosted at the gateway of the network (172.16.0.0/24) and had eight host machines in total [3].

The network was connected using SDN switch with OpenFlow v1.3 for (1) vulnerability scanning to gather knowledge about known attacks in the cloud, (2) computing the Markov Game strategy and (3) enforcing a particular deployment strategy and switching to a new one after a fixed time period T. For the first step we used scanning tools like OpenVAS, as described earlier. For the second step, we use our strategy computation algorithm that also solves the optimization problem using Gurobi solver. For the last step, we used enable and disable (kill) scripts for the different IDS mechanisms and perform them using SDN protocols.

Results. In the formulated Markov Game, we had eight states corresponding to each VM in our cloud system. In figure 6, we highlight the defender's value for state s_1. The state s_1 (attacker has root privilege on the Firewall VM in this state) was a lucrative vantage point for an attacker in the network because all other states were reachable via an array of vulnerabilities accessible in s_1. The defender had pure strategies in this state that deployed a set of IDS mechanisms (as opposed to a single IDS). These pure strategies detected a particular set of vulnerabilities and were either easier or profitable (in terms of resource usage) to deploy together. For example, deployment of two Network-based IDS using Snort is easier to configure for the Admin than the deployment of a host-based and a network-based IDS at the same time.

Host	High	Medium	Low
172.16.0.22	4	14	1
172.16.0.23	2	3	1
172.16.0.8	3	8	3
172.16.0.16	0	13	6
172.16.0.20	0	2	1
172.16.0.11	0	1	2
172.16.0.1	0	0	1
172.16.0.21	0	0	1
Total– 8	9	41	16

Fig. 5. The number of vulnerabilities in our cloud system found via persistent exploration with OpenVAS vulnerability scanner.

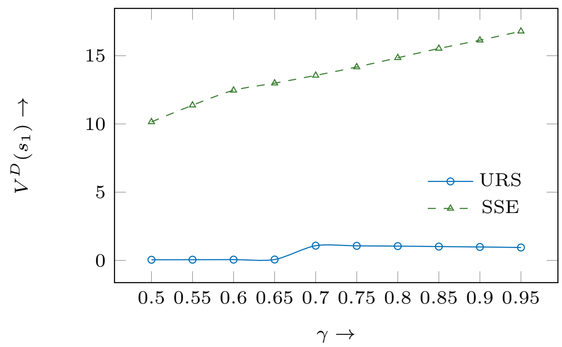

Fig. 6. Defender's value for the state s_1 as discount factor increases from 0.5 to 1.

For this study, we did not consider a high negative utility in the terminal state. We also noticed that the magnitude of positive defender utility (obtained using impact score) and negative defender utility (calculated using Mini-NET simulation similar to [28]) were comparable. Thus, the defender's value for all the states (and thus, s_1) turned out to be positive. If figure 6, a key reason for the high value gains w.r.t. URS is because URS neither paid more attention to attacks that had higher impact nor cared about the performance, both of which were essential given s_1 is a critical vantage point in the cloud network.

5 Related Work

Sheyner *et al* [12] presented a formal analysis of attacks on a network with cost-benefit analysis that suffers from scalability issues in the context of cloud networks. To solve this problem, authors in [4] provided a polynomial-time method for attack-graph construction using distributed algorithms. However, these methods model the security situation from an attacker's point of view and are mostly used to characterize the qualitative impact of an attack. We go beyond this notion and leverage this representation to construct a two-player game that helps the defender to come up with strategic defenses.

Authors in [13] introduced the idea of moving secret proxies to new network locations using a greedy algorithm, which they show can thwart brute force and DDoS attacks. In [35], Zhuang *et al* showed that the MTD system designed with intelligent adaptations improved the effectiveness further. In [29] authors show that intelligent strategies based on common intuitions can be detrimental to security and highlight how game theoretic reasoning can alleviate the problem. On those lines, Wei *et al* [19] and Sengupta *et al* [28, 27] use a game theoretic approach to model the attacker-defender interaction as a two-player game where they calculate the optimal response for the players using Nash and the Stackelberg Equilibrium concepts. The flavor of the approaches are similar to those of

Stackelberg Security Games (SSGs) for numerous physical security applications highlighted in [23, 31]. Although they talk about the use of the Markov Decision Process (MDP) approaches for MTD, they leave it as future work. On these lines, authors in [33], show that the SSE of markov games can be arbitrarily sub-optimal for stocastic discouted path games and provide an LP approximation for calculating the former. In this work, we believe that the Markovian assumption is sufficient to capture the strategy of an attacker and propose a dynamic programming based anytime solution method to find the SSE.

In the context of cloud systems, [25] discussed a risk-aware MTD strategy where they modeled the attack surface as a non-decreasing probability density function and then estimated the risk of migrating a VM to a replacement node using probabilistic inference. In [14], authors highlight obfuscation as a possible MTD strategy in order to deal with attacks like OS fingerprinting and network reconnaissance in the SDN environment. Furthermore, they highlight that the trade-off between such random mutations, which may disrupt any active services, require analysis of cost-benefits. In this work, we follow suit and consider the trade-off between security and performance of the cloud system.

6 Conclusion and Future Work

A cloud network is composed of heterogeneous network devices and applications interacting with each other. The interaction of these entities over a complex and hierarchical network structure poses a substantial risk to the overall security of the cloud system. At the same time, it makes the problem of monitoring ongoing attacks by adversaries located both outside and inside the network a challenging problem. In this paper, we model the concept of Moving Target Defense (MTD) for shifting the detection surface in the cloud system as a Markov Game. This helps us reason about (1) the security impact of multi-stage attacks that are characteristics of Advanced Persistent Threats (APT) while (2) ensuring that we do not place all possible security mechanisms in the cloud system, thereby hogging all the valuable cloud resources. The various parameters of our Markov Game are obtained using an array of softwares, prior research and publicly available metrics ubiquitous in the security community. We propose a dynamic programming based anytime algorithm to find the Strong Stackelberg Equilibrium of our Markov Game and highlight its superiority to baseline strategies for an emulated and a small real-world cloud network setup.

Acknowledgment We want to thank the reviewers for their comments. This research is supported in part by these research grants: Naval Research Lab N00173-15-G017, AFOSR grant FA9550-18-1-0067, the NASA grant NNX17AD06G, ONR grants N00014-16-1-2892, N00014-18-1-2442, N00014-18-12840, NSF-US DGE-1723440, OAC-1642031, SaTC-1528099, 1723440 and NSF-China 61628201 and 61571375. The first author is also supported by an IBM Ph.D. Fellowship.

References

1. National vulnerability database. https://nvd.nist.gov, accessed: 2018-09-25
2. Basak, A., Černỳ, J., Gutierrez, M., Curtis, S., Kamhoua, C., Jones, D., Bošanskỳ, B., Kiekintveld, C.: An initial study of targeted personality models in the flipit game. In: International Conference on Decision and Game Theory for Security. pp. 623–636. Springer (2018)
3. Chowdhary, A., Dixit, V.H., Tiwari, N., Kyung, S., Huang, D., Ahn, G.J.: Science dmz: Sdn based secured cloud testbed. In: IEEE Conference on Network Function Virtualization and Software Defined Networks (2017)
4. Chowdhary, A., Pisharody, S., Huang, D.: Sdn based scalable mtd solution in cloud network. In: ACM Workshop on Moving Target Defense (2016)
5. Chowdhary, A., Sengupta, S., Huang, D., Kambhampati, S.: Markov game modeling of moving target defense for strategic detection of threats in cloud networks. AAAI Workshop on Artificial Intelligence for Cyber Security (2019)
6. Chung, C.J., Khatkar, P., Xing, T., Lee, J., Huang, D.: Nice: Network intrusion detection and countermeasure selection in virtual network systems. IEEE transactions on dependable and secure computing **10**(4), 198–211 (2013)
7. Competition, W.R.C.D.: WRCCDC. https://archive.wrccdc.org/images/2018/ (2018)
8. Enterprises, N.: Nagios (2015)
9. Guerrero, D., Carsteanu, A.A., Huerta, R., Clempner, J.B.: An iterative method for solving stackelberg security games: A markov games approach. In: Electrical Engineering, Computing Science and Automatic Control (CCE), 2017 14th International Conference on. pp. 1–6. IEEE (2017)
10. Houmb, S.H., Franqueira, V.N., Engum, E.A.: Quantifying security risk level from cvss estimates of frequency and impact. JSS **83**(9), 1622–1634 (2010)
11. Jajodia, S., Park, N., Serra, E., Subrahmanian, V.: Share: A stackelberg honey-based adversarial reasoning engine. ACM Transactions on Internet Technology (TOIT) (2018)
12. Jha, S., Sheyner, O., Wing, J.: Two formal analyses of attack graphs. In: Computer Security Foundations Workshop, 2002. Proceedings. 15th IEEE. pp. 49–63. IEEE (2002)
13. Jia, Q., Sun, K., Stavrou, A.: Motag: Moving target defense against internet denial of service attacks. In: 2013 22nd International Conference on Computer Communication and Networks. pp. 1–9. IEEE (2013)
14. Kampanakis, P., Perros, H., Beyene, T.: Sdn-based solutions for moving target defense network protection. In: IEEE 15th International Symposium on a World of Wireless, Mobile and Multimedia Networks. IEEE (2014)
15. Korzhyk, D., Conitzer, V., Parr, R.: Complexity of computing optimal stackelberg strategies in security resource allocation games. In: AAAI (2010)
16. Korzhyk, D., Yin, Z., Kiekintveld, C., Conitzer, V., Tambe, M.: Stackelberg vs. Nash in Security Games: An Extended Investigation of Interchangeability, Equivalence, and Uniqueness. J. Artif. Int. Res. **41**(2), 297–327 (May 2011), http://dl.acm.org/citation.cfm?id=2051237.2051246
17. Korzhyk, D., Yin, Z., Kiekintveld, C., Conitzer, V., Tambe, M.: Stackelberg vs. nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. Journal of Artificial Intelligence Research **41**, 297–327 (2011)

18. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: Eleventh International Conference on Machine Learning (1994)
19. Lye, K.W., Wing, J.M.: Game strategies in network security. International Journal of Information Security (2005)
20. McCumber, J.: Information systems security: A comprehensive model. In: Proceedings of the 14th National Computer Security Conference (1991)
21. Miehling, E., Rasouli, M., Teneketzis, D.: Optimal defense policies for partially observable spreading processes on bayesian attack graphs. In: Proceedings of the Second ACM Workshop on Moving Target Defense. pp. 67–76. ACM (2015)
22. Nguyen, T.H., Wright, M., Wellman, M.P., Singh, S.: Multistage attack graph security games: Heuristic strategies, with empirical game-theoretic analysis. Security and Communication Networks **2018** (2018)
23. Paruchuri, P., Pearce, J.P., Marecki, J., Tambe, M., Ordonez, F., Kraus, S.: Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In: AAMAS, 2008. pp. 895–902 (2008)
24. Paruchuri, P., Pearce, J.P., Marecki, J., Tambe, M., Ordonez, F., Kraus, S.: Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In: AAMAS (2008)
25. Peng, W., Li, F., Huang, C.T., Zou, X.: A moving-target defense strategy for cloud-based services with heterogeneous and dynamic attack surfaces. In: IEEE International Conference on Communications (ICC) (2014)
26. Schlenker, A., Thakoor, O., Xu, H., Fang, F., Tambe, M., Tran-Thanh, L., Vayanos, P., Vorobeychik, Y.: Deceiving cyber adversaries: A game theoretic approach. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. pp. 892–900. International Foundation for Autonomous Agents and Multiagent Systems (2018)
27. Sengupta, S., Chakraborti, T., Kambhampati, S.: Mtdeep: Boosting the security of deep neural nets against adversarial attacks with moving target defense. In: Workshop on Engineering Dependable and Secure Machine Learning Systems. AAAI (2018)
28. Sengupta, S., Chowdhary, A., Huang, D., Kambhampati, S.: Moving target defense for the placement of intrusion detection systems in the cloud. In: International Conference on Decision and Game Theory for Security. pp. 326–345. Springer (2018)
29. Sengupta, S., Vadlamudi, S.G., Kambhampati, S., Doupé, A., Zhao, Z., Taguinod, M., Ahn, G.J.: A game theoretic approach to strategy generation for moving target defense in web applications. AAMAS (2017)
30. Shapley, L.S.: Stochastic games. Proceedings of the national academy of sciences **39**(10), 1095–1100 (1953)
31. Sinha, A., Nguyen, T.H., Kar, D., Brown, M., Tambe, M., Jiang, A.X.: From physical security to cybersecurity. Journal of Cybersecurity **1**(1), 19–35 (2015)
32. Venkatesan, S., Albanese, M., Cybenko, G., Jajodia, S.: A moving target defense approach to disrupting stealthy botnets. In: Proceedings of the 2016 ACM Workshop on Moving Target Defense. pp. 37–46. ACM (2016)
33. Vorobeychik, Y., Singh, S.: Computing stackelberg equilibria in discounted stochastic games (corrected version) (2012)
34. Zhuang, R., DeLoach, S.A., Ou, X.: Towards a theory of moving target defense. In: ACM MTD Workshop, 2014. pp. 31–40. ACM (2014)
35. Zhuang, R., Zhang, S., Bardas, A., DeLoach, S.A., Ou, X., Singhal, A.: Investigating the application of moving target defenses to network security. In: 6th International Symposium on Resilient Control Systems (ISRCS). IEEE (2013)