SDN-based Moving Target Defense using Multi-agent Reinforcement Learning

Ankur Chowdhary[1], Dijiang Huang[1], Abdulhakim Sabur[1], Neha Vadnere[1], Myong Kang[2], and Bruce Montrose[2]

Arizona State University, USA[1]
US Naval Research Lab, USA[2]
{achaud16,dijiang,asabur,nvadnere}@asu.edu
{myong.kang,bruce.montrose}@nrl.navy.mil

Abstract. The static nature of network defenses provides an asymmetric advantage to the attacker. Moving Target Defense (MTD) aims at continuously shifting the underlying infrastructure to present an inaccurate view of the network to the attacker, and increase the complexity of mounting attacks against the network. An adaptive adversary may, however, understand the MTD over time. Hence the defense strategy needs to continuously evolve defense and adapt to the emerging threats. The defense strategy also needs to consider the dynamic nature of the network infrastructure. The defense strategy should have minimal impact on service availability. In this research work we present a Multi-agent reinforcement learning framework that model's attacker and defender's interactions as a dynamic multi-stage game in a purely adversarial setting in a Software-defined Network (SDN) managed cloud environment. The game-theoretic model accounts for uncertainty associated with the state of the underlying network when deploying optimal MTD. Our empirical evaluation shows better reward for defenders when using a reinforcement learning policy against an adaptive adversary in different game-theoretic settings.

Keywords: Multi-agent Reinforcement Learning, Moving Target Defense (MTD), Deep-Q Learning (DQN), Software-defined Networks (SDN), Markov Game

1 Introduction

Traditional defense mechanisms suffer from many shortcomings. First, the attacker has enough time to scan the network infrastructure for security vulnerabilities. Second, the defense mechanisms such as Firewall, Intrusion Detection and Prevention Systems (IDPS) are signature-based. Hence, they can identify and stop only known attacks. The goal of Moving Target Defense (MTD) is to increase the attack's complexity by continuously moving the components of the underlying system. MTDs not only increase the uncertainty for the attacker, but also disrupt the chain of attack carefully crafted by adversaries for targeting unknown attacks like Advanced Persistent Threat (APT) [1]. One key challenge that limits the adoption of MTD in a Software-defined Network (SDN) environment is the impact of change in network configuration on the system performance and usability, as discussed by Sengupta *et. al.* [18]. The impact of the dynamic placement of NIDS on network latency has been considered in [22] and [16]. The performance cost associated with MTD is considered as utility values in a repeated game setting by research works [3,13]. Software-defined Networking (SDN) [10] provides centralized monitoring and orchestration for the network. SDN has emerged as a state-of-the-art network architecture for data centers and backbone networks. SDN decouples data plane and control plane to provide key network functions such as routing algorithm and load-balancing using programmable APIs. In this paper we model attacker and defender's interactions as a two-player game. We consider that both attacker and defender learn each other's strategy over time. The defender has a fully observable environment, and defender can deploy different MTD countermeasures to deal with attacker's actions such as reconnaissance, targeted attacks, etc. We use Common Vulnerability Scoring System (CVSS) [14] metrics such as *impact score*, and *access complexity* for defining the rewards obtained by attacker and defender based on their actions as discussed by Chowdhary *et. al.* [4]. We use multi-agent reinforcement learning framework based on Deep-Q Learning (DQL) [24] to model the attacker's and defender's actions. Each MTD action can have some impact on network services, such as availability impact, conflict of security rule deployed as part of MTD with existing security rules [12]. Some research works consider the performance impact induced by defensive countermeasures as a penalty on game-theoretic model [16]. Based on similar lines, we incorporate the policy conflict scenarios in our game-theoretic model. In summary the key contribution of this research work are as follows: *a)* A zero-sum dynamic game formulation between the attacker and defender, considering the impact of defense countermeasures (policy conflict scenarios') as part of reward modeling. We introduced domain-specific reward modeling, i.e., the model considers the CVSS score of vulnerability, difficulty of compromising a vulnerability (CVSS access complexity), attacker and defenders' effort, and effect of MTD countermeasures. *b)* A multi-agent reinforcement learning (MARL) scheme for obtaining the optimal Moving Target Defense (MTD) strategy. The MARL formulation accounts for intelligent attackers' (who learn defenders' strategy over time), and provides an optimal reinforcement learning solution to defend against adaptive adversaries in a competitive game. The optimal defender obtained using reinforcement learning

2 Related Work

2.1 SDN-based Moving Target Defense (MTD)

The *cost-benefit* analysis of MTD adaptations against network mapping attempts has been discussed by [9]. The SDN-based MTD can introduce various countermeasures at the network-level such as network shuffling [7], route modification [21], IP and port obfuscation as discussed by Wang *et al.* [20]. Chowdhary *et al.* [3] utilized SDN-environment to analyze the hosts mounting DDoS attacks on critical network services. The SDN-controller downgrades network bandwidth (how to switch) using a *Nash-Folk* theorem-based punishment mechanism. Jafarian *et al.* [8] use OpenFlow based random host mutation technique to switch virtual IP (what to switch) targeted by reconnaissance attempts. This work propose considers how to mutate IP address with a high degree of unpredictability while keeping a stable network configuration and minimal operational overhead. Debroy *et. al.* [5] used the SDN framework to identify the optimal migration rate (when to switch) and ideal migration location for VMs under the DoS attack.

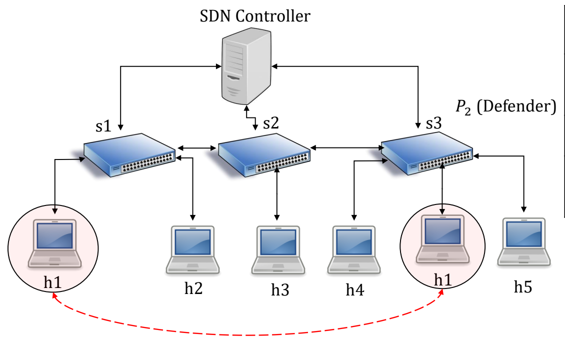
2.2 MTD as a Game-Theoretic Model

A multi-agent partially observable Markov Decision Process model for MTD has been proposed by Eghtesad *et. al.* [6] The research work considers Deep-Q Learning [24] and Double Oracle (DO) algorithms to model attacker-defender interactions as two-player games. However, the reward model and action space are quite simplistic in this research work. Only *control/disrupt* actions are considered for an attacker and *confidentiality/availability* for the defender. The binary reward values used in modeling do not represent the actual threat level given the current network setup. A mullti-agent reinforcement learning solution using Bayesian Stackelberg Markov Games (BSMGs) was introduced by Sengupta *et. al.* [19]. The framework models uncertainty over attacker types and MTD systems. This research uses domain-specific reward metrics based on the Common Vulnerability Scoring System (CVSS) [14]. The framework provides optimal MTD for web applications, proposed Bayesian Strong Stackelberg Q-Learning (BSS-Q) algorithm converges to optimal policy for MTD domains with incomplete information.

3 Background

In this section we consider the threat model and describe the background concepts associated with the game-theoretic formulation. Consider the attacker has access to the end-hosts h1-h5. The attacker's objective is to obtain information about the software running on the end hosts (e.g., web server, SQL database). The attacks aimed at discovering network topology and software vulnerabilities are generalized as 'Reconnaissance (Recon)' attacks (see Figure 1). The information obtained from the *Recon* phase is used by the attacker to mount attacks such as SQL Injection (SQLI), Cross-site scripting (XSS). We generalize these attacks as 'Exploitation'. The consequence of these attacks is information disclosure and loss of data.

The defender can use Moving Target Defense (MTD) techniques to delay the attack progress. Some MTD countermeasures we considered include Shuffle (host VM migration), and IP mutation (dynamic IP address reassignment). The use of these countermeasures can, however, lead to violation of security policies.



P_1 (Attacker)

	No Action	Recon	Exploitation
No Action	0,0	$3 - f(C_{A,}C_D)$, $f(C_{A,}C_D) - 3$	$7 - f(C_{A,}C_D)$, $f(C_{A,}C_D) - 7$
Shuffle	$f(C_D)$, $-f(C_D)$	$3 - f(C_{A,}C_D)$, $f(C_{A,}C_D) - 3$	$7 - f(C_{A,}C_D)$, $f(C_{A,}C_D) - 7$
IP Mutation	$f(C_D)$, $-f(C_D)$	$3 - f(C_{A,}C_D)$, $f(C_{A,}C_D) - 3$	$3 - f(C_{A,}C_D)$, $f(C_{A,}C_D) - 3$

Security Policies: [r1: 10.1.0.1 (h1) → 10.1.0.2 (h2), ALLOW]
[r2: 10.1.0.0/24 (h1,h2) → 10.3.0.0/24 (h4,h5), DENY]
MTD Countermeasure: [VM Migration: (h1) → 10.3.0.3]
r3: 10.3.0.3 (h1) → 10.1.0.2 (h2), ALLOW]
r2 conflicts with r3, header match, different actions.

Fig. 1: SDN-managed cloud network with two-player zero-sum game formulation

Consider, the deployment MTD countermeasure IP mutation. If the host targeted by the attacker is h1, with IP address 10.1.0.1, we have a rule r1 which allows traffic between hosts h1 and h2 (IP address 10.1.0.2). Another rule r2 prohibits any traffic between subnet under switch s1 (10.1.0.0/24) and subnet under switch s3 (10.3.0.0/24). As a consequence of MTD, the host h1 is migrated to subnet under switch s3 and gets reassigned IP address 10.3.0.3. In order to maintain service availability between hosts h1 (new location) and h2, a new rule r3 is added. This rule violates the security requirement of rule r2. The concept of security policy conflict has been discussed in detail by Pisharody *et. al.* [12]. The conflicts can be resolved in an automated fashion or using manual analysis, but this requires additional effort on part of defender. Thus, we consider the consequence of MTD in our game theoretic formulation. Next, we discuss the game-theoretic model.

3.1 Game Theoretic Model

We consider two players in set P, Attacker (P_1) and the Defender (P_2) represented in the game-theoretic formulation. The game theoretic model can be defined using tuple {P, S, A, R, τ, γ}. The variables players (P), state (S), actions (A), reward (R), transition function (τ), discount factor (γ). The CVSS system is well-known for tracking known vulnerabilities. We use CVSS metrics for domain-specific reward modeling in this research work. Each vulnerability is assigned a score that determines the severity of the vulnerability. Moreover, there is metadata associated with each vulnerability, such as Access Complexity (AC), which determines how easy or difficult it is to exploit it. For instance, if AC is low, it is easier to exploit the vulnerability. The range of CVSS score is $(0, 10]$, and AC = {`LOW, MEDIUM, HIGH`}.

Actions The players have actions defined using A $= \{a_1, a_2\}$, attacker and defenders' actions. We consider that attacker can choose from actions $a_1 = \{$No Action, Recon, Exploitation$\}$, and defender can choose from actions $a_2 = \{$No Action, Shuffle, IP Mutation$\}$ (see Figure 1).

States The state of game is defined in terms of privilege of each player. The state transition depends on the actions of both the players. Suppose in the current state of the game, and the attacker is at s_0=user. If we consider the normal form of the game, actions of players are a_1^1=No Actions, a_1^2= Exploitation, the attacker will be able to compromise a given vulnerability, with transition probability defined by $\tau \in (0, 1]$. As a consequence, the state of the attacker will be s_1=Root. The transition probability will depend on the access complexity of the security vulnerability as discussed by Chowdhary *et. al.* [4]. $\tau = \{s_0, a_1^1 \times a_2^1, s_1\}$.

Transition Probabilities $\tau(s_0, a_1^1, a_2^1, a_1)$ represent the state transition probability. By taking action a_1^1 in state s_0 the attacker can transition to s_1 with a certain probability, provided that the defender takes the action s_2^1. The transition probability depends on how difficult it is to exploit a certain vulnerability, and the action taken by defender. For instance if s_0 = 'user', it is easy to exploit a vulnerability a_1^1 = 'Exploitation' and defender has no monitoring in place, i.e., a_2^1 = 'No Action', the attacker moves to state s_1 = 'root' with a high probability, or ($\tau = 0.9$). We use access complexity as a measure of how easy it is to discover or exploit a vulnerability.

Rewards We capture the rewards in terms CVSS score of the security vulnerability, cost of deploying defense, and impact of the MTD countermeasure on the state of the security policies (security policy violation leads to negative reward). We consider reward as a function of cost of attack C_A, the cost associated with defense C_D. The cost of defense can be further considered as a cost of deploying defensive countermeasure, and the cost of policy conflict management associated with any security policy conflicts introduced by countermeasure such as firewall rule. An example to illustrate this can be seen in Figure 1. The migration of host (h1) to a different subnet creates a case of security policy conflict. thus utility (reward) for the attacker R_1^1 can be expressed as $3 - f(C_A, C_D)$, where 3.0 is the CVSS score of security vulnerability, and utility for the defender R_2^1 can be expressed as $f(C_A, C_D) - 3$.

3.2 Optimal Policy in a Markov Game

The goal of each player i is to optimize its own long term reward, by finding the policy $\pi^i \rightarrow \Delta(A_i)$. As a result the value function $V^i : S \rightarrow R$ becomes a function of joint policy of both agents $\pi : S \rightarrow \Delta(A)$. The function can be defined as $\pi(A|S) := \Pi_{i \in N} \pi_i(A_i, S)$. The joint policy can be expressed as

$$V^i_{\pi^i, \pi^{-i}} = E[_{t \geq 0} \gamma^t R^i(s_t, a_t, s_{t+1}) | a_t^i \sim \pi^i(.|s_t), s_0 = s] \quad (1)$$

The -i represents the indices of all agents except agent i. The optimal performance of an agent in a Markov Game setting is controlled not only by its policy, but also by other players' policies.

Nash Equilibrium in a Markov Game The Nash Equilibrium (NE) is the joint policy $\pi^* = (\pi^{1,*}, ..., \pi^{N,*})$ for each agent such that they do not obtain higher incentive by deviating from policy π^*. The value function for NE can be expressed as

$$V^i_{\pi^{i,*},\pi^{-i,*}}(s) \geq V^i_{\pi^i,\pi^{-i,*}}(s) \quad \forall \pi^i \tag{2}$$

For the agent $i \in N$, policy obtained during NE $\pi^{i,*}$ is best response to $\pi^{-i,*}$. As noted by Sengupta *et. al.* [17] NE exists for discounted general-sum Markov Games, but it may not be unique in general. The calculation of equilibrium is relatively straightforward in a zero-sum Markov Game setting as demonstrated by Chowdhary *et. al.* [4]. The complexity of the value iteration algorithm used is polynomial in terms of the number of states and actions.

4 Multi-Agent Reinforcement Learning Model

Reinforcement Learning (RL) consists of a single agent's interaction with the environment. The agent train's over multiple iterations (epochs) and aims at obtaining the best long term return. During each epoch, the agent observes the states$s$, takes action a, and receives the reward R. The agent moves to state s' on taking action a.

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} \tau(s,a,s') \quad max'_a Q^*(s',a') \tag{3}$$

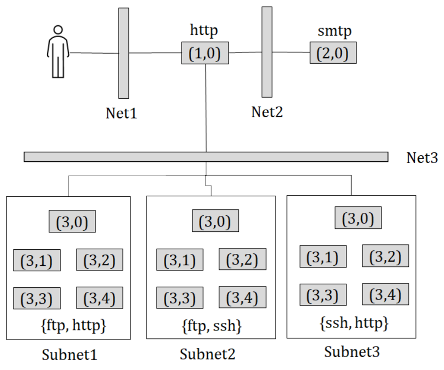
We employ a well-known model-free reinforcement learning algorithm Q-learning [23] to learn the agent's policy. Q-Learning is a popular mechanism of incrementally estimating the utility values of executing an action in a given state. The Q-values are continuously updated based on the equation. The value γ is a discount factor that characterizes the agent's preference for future reward compared to immediate reward. The Q-learning shown in the equation above helps in find an optimal policy to maximize the expected reward value. However, it suffers from a lack of generality. The Q-learning agent will not be able to determine the action for states it has not seen.

To deal with this situation, we utilize Deep-Q Network (DQN) [24], a neural network-based solution to estimate the Q-value function. The approximate value function is parameterized as $Q(s, a; \theta_i)$ using a deep convolutional neural network, where θ_i represents the Q-network model parameters at iteration i. Q-network follows an experience replay approach, where agent's experience $e_t = \{s_t, a_t, r_t, s_{t+1}\}$, are stored in the dataset $D_t = \{e_1, e_2, .., e_t\}$. During the learning phase, the Q-learning updates are applied to experience samples drawn uniformly and randomly from a sample pool [11].

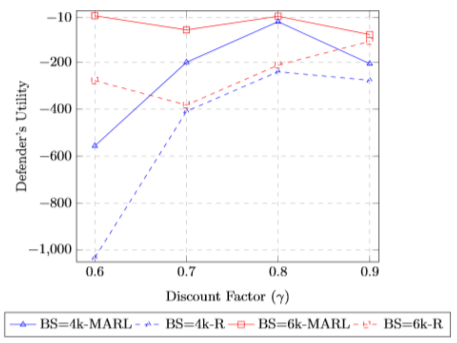
In Multi-agent Reinforcement Learning (MARL) the agent interacts with the environment and other agents in a sequential game. Each agents aims at optimizing its long term return. The MARL can be placed into different settings, a *fully cooperative*, *fully competitive*, and a *mix of the two* [25]. Each agent has limited information about the observation of other agents, leading to suboptimal local decisions. Two popular representative frameworks used for the representation of MARL are Markov/stochastic games, and extensive form games as noted by Zhang *et. al.* [25]. We consider the MARL setting for the interaction between attacker and defender in a fully competitive setting. The Nash Equilibrium (NE) discussed in the previous section can serve as a model for robust learning in a competitive setting.

5 Experimental Evaluation

We simulated an enterprise network with an industrial control system (Subnet2), and IoT devices (Subnet3). The network comprised of 16 hosts, three networks (Net1-3). A mixture of Windows and Linux systems were utilized for experimental simulation. The network consists of four key services (SSH, FTP, HTTP, and SMTP), as shown in Figure 2a. The network scan for obtaining CVSS score was performed using standard network vulnerability scanning tools. We utilized the network configuration, vulnerability parameters such as CVSS score, access complexity to create a transition, and reward matrices. The attack vectors used for empirical evaluation comprised reconnaissance, and vulnerability exploitation, whereas the defender used approaches such as shuffle, and IP mutation as part of his moving target defense (MTD) strategy.



(a) Network Setup with multiple set of vulnerabilities. The defense mechanism IP shuffle and service migration was used by the defender



(b) Defender's utility with different values of γ, and batch sizes (4k,6k). The utility using MARL formulation with defender playing mixed strategy is greater than random strategy (R)

We utilized OpenAI Gym [2] for formulation of Markov Game between the attacker and defender. A multi-agent reinforcement learning solver with Deep-Q Network (DQN) was used for the evaluation of optimal policy for the defender. We compared the defender's utility for different settings of multi-agent reinforcement learning strategy (see Figure 2b). The negative value indicates the cost induced by defense investment which is always negative in our zero-sum game formulation. In MARL-4k and MARL-6k both attacker and defender use reinforcement learning strategies in Markov game formulation, whereas in the case of random strategies, R-4k, R-6k, the defender uses random strategy against attacker's reinforcement learning strategy. It can be observed that the defender obtains a higher reward using the reinforcement learning strategy, -200 for $\gamma = 0.7$, compared to -409 when playing random strategy. The utility obtained for batch size 6k is higher than batch size of 4k, for different values of the discount factor. The best utility for a defender is obtained when the discount factor is 0.8 (-27 for batch size 4k, and -4 for batch size 6k). The utility value for the defender increases with a discount factor up to value $\gamma = 0.8$, but decreases after that, which indicates that the defender needs to evaluate his strategy with more emphasis on MTD strategy deployed for future states. Thus, the empirical evaluation suggests that the defender can mimic the attacker's action using a reinforcement learning policy. Thus, it comes up with an MTD strategy producing higher utility instead of using a random MTD strategy to play all actions with equal probability from the given action set.

6 Discussion

6.1 Continuous Model Evaluation

There are several challenges associated with the use of reinforcement learning in the cybersecurity domain. The structure of network and services keeps on changing in a network; hence, the number of states is rarely stationary. This poses a challenge of continuous model update for the defender if they employ a reinforcement learning model as part of their detection system. Another key challenge in the MARL is the non-stationary environment induced by the actions of multiple agents learning simultaneously. In order to handle the issue, we assume each agent learns independently with the goal of optimizing its policy.

6.2 Reward Formulation

Our game-theoretic framework's reward structure also depends on CVSS score of vulnerabilities present, cases of policy conflict and cost of attack/defense. Thus we consider a factorized reward structure in our MARL network design. This also helps address the state-space explosion problem associated with the joint action space of two-player (combinatorial nature of MARL). Exiting MTD research works [15], [22] have selected random values for attacker/defender utility and cost. In this research we perform domain-specific reward modeling. The attacker's utility is a function of CVSS score, consequences of MTD countermeasures (security policy conflicts [12]), and attack/defense costs.

7 Conclusion

We presented a moving target defense (MTD) solution based on zero-sum Markov game in this research work. We utilized a Multi-agent Reinforcement Learning (MARL) based solution to evaluate the defender's utility. We performed a domain specific reward modeling to capture cost of MTD countermeasures in terms of security investment, and policy conflicts introduced by MTD countermeasures. We observed that the defender obtains higher utility when using a reinforcement learning approach against an adaptive attacker, than a strategy of randomly deploying MTD. As an extension of this work, we plan to perform an empirical evaluation on scalable infrastructure to establish the generalizability of the solution.

References

1. Alshamrani, A., Myneni, S., Chowdhary, A., Huang, D.: A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. IEEE Communications Surveys & Tutorials **21**(2), 1851–1877 (2019)
2. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym. arXiv preprint arXiv:1606.01540 (2016)
3. Chowdhary, A., Pisharody, S., Alshamrani, A., Huang, D.: Dynamic game based security framework in sdn-enabled cloud networking environments. In: Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization. pp. 53–58 (2017)
4. Chowdhary, A., Sengupta, S., Huang, D., Kambhampati, S.: Markov game modeling of moving target defense for strategic detection of threats in cloud networks. arXiv preprint arXiv:1812.09660 (2018)
5. Debroy, S., Calyam, P., Nguyen, M., Stage, A., Georgiev, V.: Frequency-minimal moving target defense using software-defined networking. In: 2016 International Conference on Computing, Networking and Communications (ICNC). pp. 1–6. IEEE (2016)
6. Eghtesad, T., Vorobeychik, Y., Laszka, A.: Adversarial deep reinforcement learning based adaptive moving target defense (2020)
7. Hong, J.B., Yoon, S., Lim, H., Kim, D.S.: Optimal network reconfiguration for software defined networks using shuffle-based online mtd. In: 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS). pp. 234–243. IEEE (2017)
8. Jafarian, J.H., Al-Shaer, E., Duan, Q.: Openflow random host mutation: transparent moving target defense using software defined networking. In: Proceedings of the first workshop on Hot topics in software defined networks. pp. 127–132 (2012)
9. Kampanakis, P., Perros, H., Beyene, T.: Sdn-based solutions for moving target defense network protection. In: World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a. pp. 1–6. IEEE (2014)
10. Kreutz, D., Ramos, F.M., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: A comprehensive survey. Proceedings of the IEEE **103**(1), 14–76 (2014)
11. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)

12. Pisharody, S., Natarajan, J., Chowdhary, A., Alshalan, A., Huang, D.: Brew: A security policy analysis framework for distributed sdn-based cloud environments. IEEE transactions on dependable and secure computing (2017)
13. Prakash, A., Wellman, M.P.: Empirical game-theoretic analysis for moving target defense. In: Proceedings of the second ACM workshop on moving target defense. pp. 57–65 (2015)
14. Scarfone, K., Mell, P.: An analysis of cvss version 2 vulnerability scoring. In: 2009 3rd International Symposium on Empirical Software Engineering and Measurement. pp. 516–525. IEEE (2009)
15. 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)
16. 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)
17. Sengupta, S., Chowdhary, A., Huang, D., Kambhampati, S.: General sum markov games for strategic detection of advanced persistent threats using moving target defense in cloud networks. In: International Conference on Decision and Game Theory for Security. pp. 492–512. Springer (2019)
18. Sengupta, S., Chowdhary, A., Sabur, A., Alshamrani, A., Huang, D., Kambhampati, S.: A survey of moving target defenses for network security. IEEE Communications Surveys & Tutorials (2020)
19. Sengupta, S., Kambhampati, S.: Multi-agent reinforcement learning in bayesian stackelberg markov games for adaptive moving target defense. arXiv preprint arXiv:2007.10457 (2020)
20. Shi, Y., Zhang, H., Wang, J., Xiao, F., Huang, J., Zha, D., Hu, H., Yan, F., Zhao, B.: Chaos: An sdn-based moving target defense system. Security and Communication Networks **2017** (2017)
21. Steinberger, J., Kuhnert, B., Dietz, C., Ball, L., Sperotto, A., Baier, H., Pras, A., Dreo, G.: Ddos defense using mtd and sdn. In: NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium. pp. 1–9. IEEE (2018)
22. 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 (2016)
23. Watkins, C.J., Dayan, P.: Q-learning. Machine learning **8**(3-4), 279–292 (1992)
24. Yang, Z., Xie, Y., Wang, Z.: A theoretical analysis of deep q-learning. In: Learning for Dynamics and Control. pp. 486–489. PMLR (2020)
25. Zhang, K., Yang, Z., Başar, T.: Multi-agent reinforcement learning: A selective overview of theories and algorithms. arXiv preprint arXiv:1911.10635 (2019)