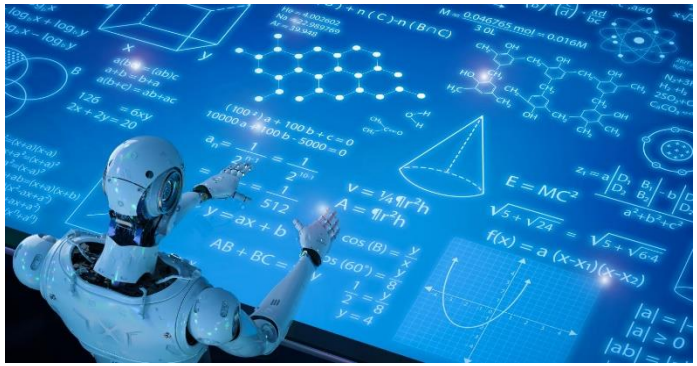
SDN-based Moving Target Defense using Multi-agent Reinforcement Learning



Ankur Chowdhary*, Dijiang Huang, Abdulhakim Sabur, Neha Vadenere, Myong Kang, and Bruce Montrose

Arizona State University

US Naval Research Lab

@drstrange1989 (achaud16@asu.edu)

1st International Conference on Autonomous Intelligent Cyber-defense Agents (AICA 2021)

Ira A. Fulton Schools of Engineering
ARIZONA STATE UNIVERSITY

INDEX

- Motivation
- SDN-based Moving Target Defense
- MTD: A multi-agent Markov Game
- Game Theoretic Model
- Optimal Policy in Markov Game
- Reinforcement Learning using Deep-Q Network (DQN)
- Multi-agent Reinforcement Learning (MARL)
- Evaluation Results
- Conclusion

Motivation

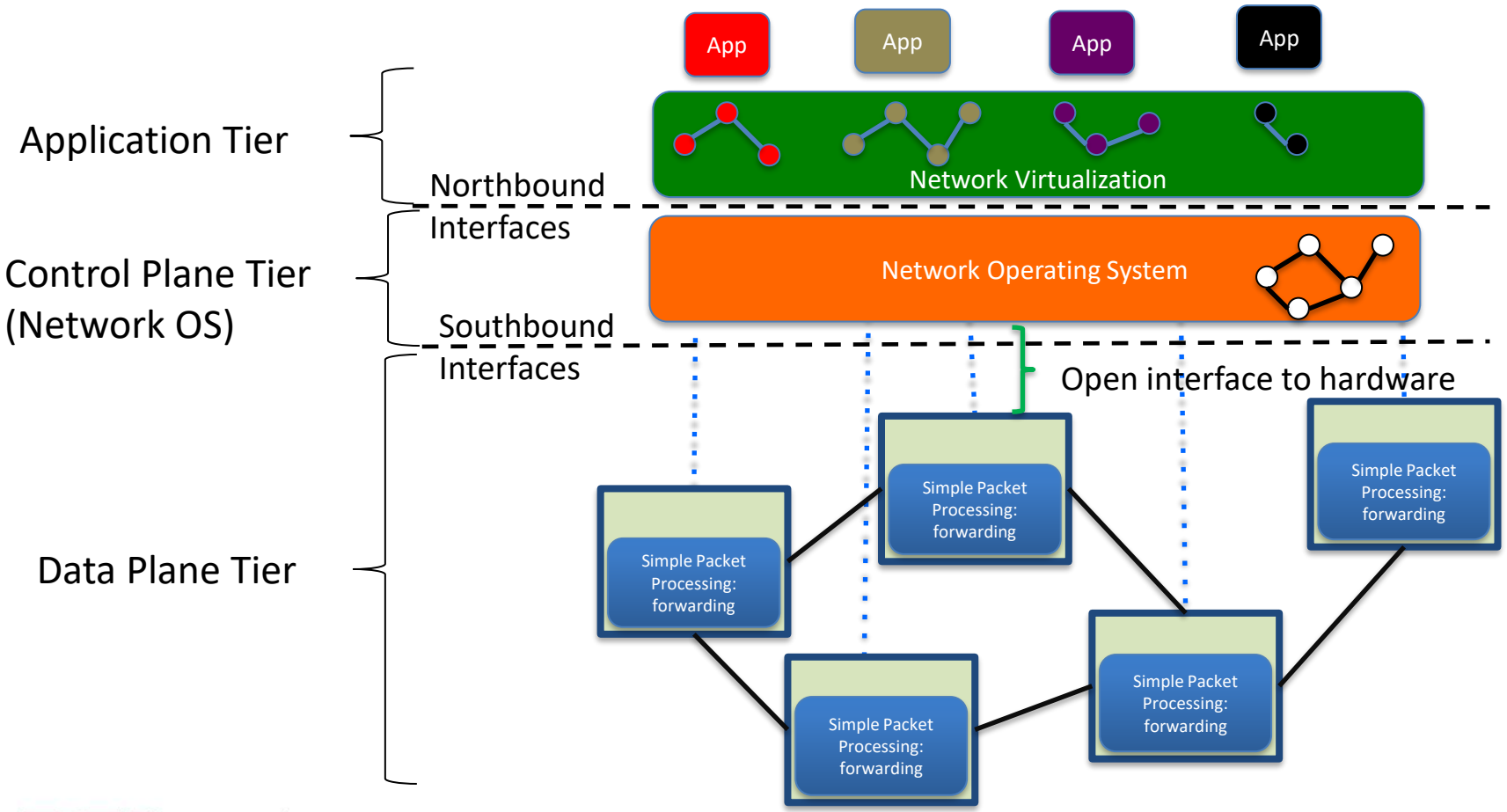
Problems

- Static nature of current network defenses.
- Modeling Attacker's and Defender's interaction.
- Adversary can be adaptive (can learn MTD).
- MTD can impact service availability.

Research Contributions

- Moving Target Defense (MTD).
- Multi-player game to model attack-defense.
- Use of reinforcement learning model.
- Reward model incorporates attack, defense, and service availability costs.

Abstract SDN Model



https://www.opennetworking.org/sdn-definition/

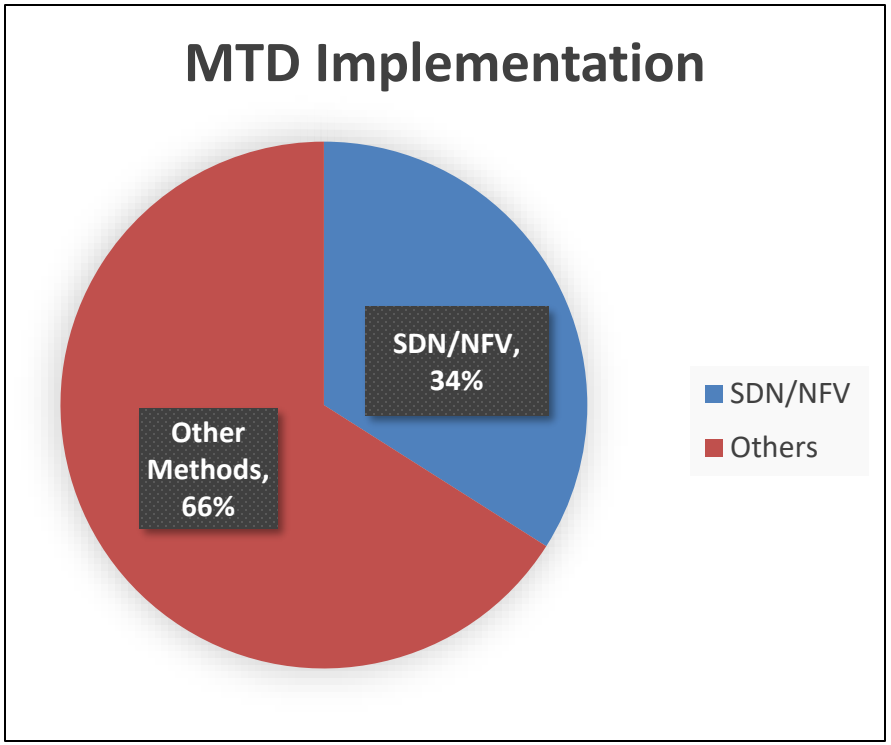
Motivation for Moving Target Defense

Static Target

vs

MOVING TARGET

SDN-based MTD



Middlebox	Misconfig.	Overload	Electric
Firewall	67.3%	16.3%	16.3%
Proxy	63.2%	15.7%	21.1%
IDS	54.5%	11.4%	34%

Table: Common causes of middlebox failures.

19. Sherry, Justine, Sylvia Ratnasamy, and Justine Sherry At. "A survey of enterprise middlebox deployments." (2012).

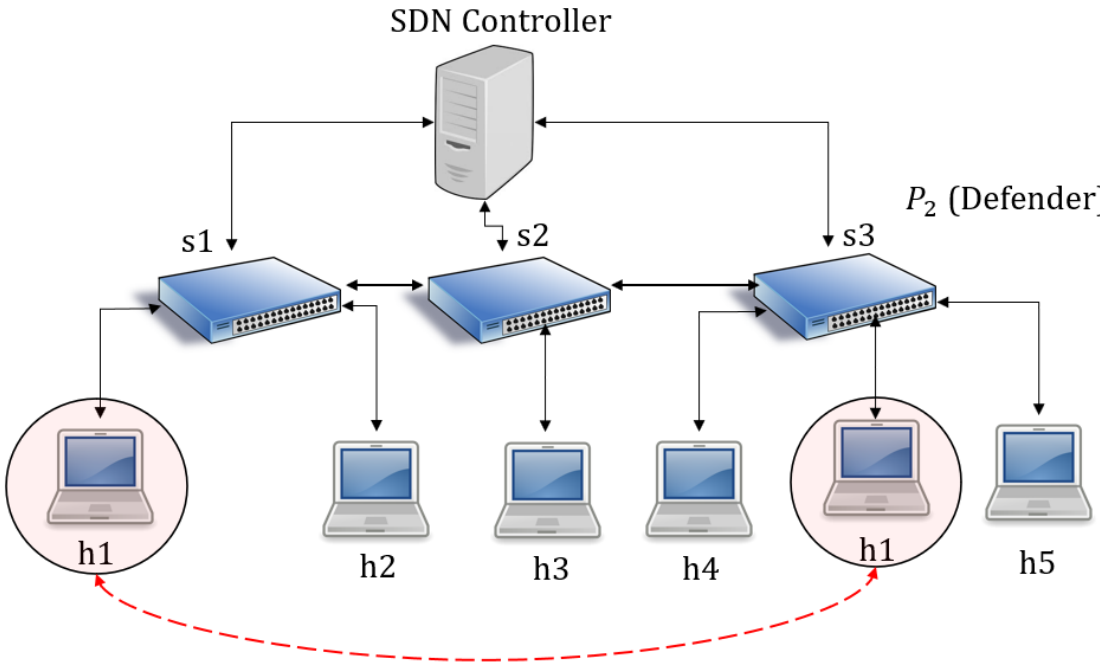
MTD: A Multi-Agent Markov Game

- Game theoretic model can be defined using players (P), states (S), actions (A), transition function (τ), discount factor (γ) {P, S, A, R, τ, γ}.
- Two players: Attacker (P_1) and Defender (P_2).
- Action sets A = $\{a_1, a_2\}$, $a_1 = \{$No Action, Recon, Exploitation$\}$, a_2 = {No Action, Shuffle, IP Mutation}.
- States S = $\{s_0, s_1\}$, $s_0 = \{$user$\}$, s_1 = {root}.

MTD: A Multi-Agent Markov Game

- Transition function $\tau = \{s_0, a_1^1 \times a_2^1, s_1\}$, where s_0 = {user}, a_1^1 = {No Action}, a_2^1 = {Exploitation}, s_1 = {root}, $\tau \in (0,1]$.
- Rewards (R) : CVSS Score – f(cost of defense C_D, cost of attack C_A).
- For e.g., if CVSS score is 3.0, $R_A = \{3.0 - f(C_A, C_D)\}$, $R_D = \{f(C_A, C_D) - 3.0\}$.

SDN-managed Cloud Environment



P_2 (Defender)

P_1 (Attacker)

	No Action	Recon	Exploitation
No Action	0,0	-3,3−C_R	-7,7−C_E
Shuffle	-2,2	-2−C_C, 2−C_R	-2−C_c, 2−C_E
IP Mutation	-1,1	-1−C_c, 1−C_R	-1−C_c, 1−C_E

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.

Optimal Policy in a Markov Game

- Each player i plans on optimizing its long-term reward, by finding policy $\pi^i \to \Delta(A_i)$.
- Value iteration function $V^i: S \to R$ aims at optimizing joint policy of both agents $\pi: S \to \Delta(A)$.
- Function is defined as $\pi(A|S) = \Pi_{i \in N}\, \pi_i(A_i, S)$.
- Joint policy can be expressed as $V^i_{\pi^i, \pi^{-i}}$

REINFORCEMENT LEARNING

- A reinforcement learning task is about training an agent which interacts with its environment.
- agent arrives at different scenarios known as states by performing actions.
- agent has only one purpose here – to maximize its total reward across an episode.
- reinforce the agent to learn to perform the best actions by experience.
- This is the strategy or policy.

Q-LEARNING

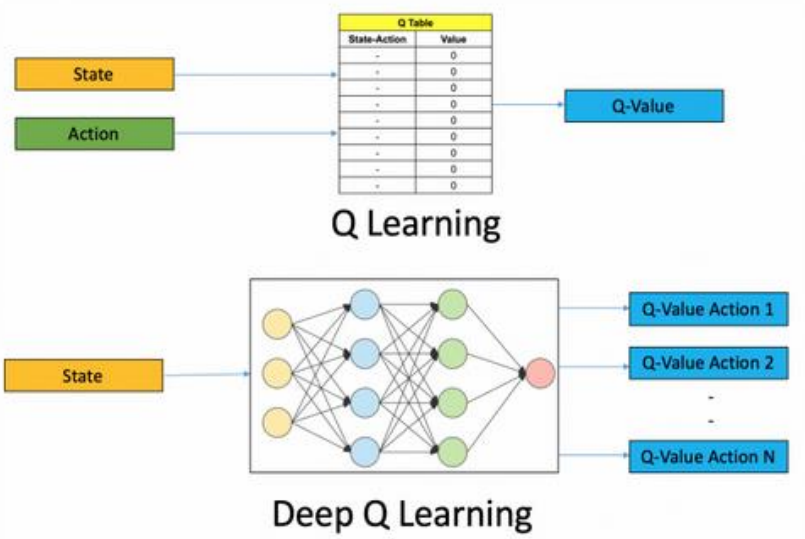
- This total reward is also called the Q-value.
- $Q(s,a) = r(s,a) + \gamma \times \max_a Q(s',a).$
- Q-value obtained from state *s* and performing action *a* is the immediate reward r(s,a) plus the highest Q-value possible from the next state *s'*.
- γ – Future rewards.
- Q(s',a) again depends on Q(s",a) which will then have a coefficient of gamma squared.
- $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \times \max_a Q(S_{t+1}, a\right) - Q(S_t, A_t)]$
- α – learning rate.

DEEP Q-LEARNING

- Imagine an environment with 10,000 states and 1,000 actions per state. This would create a table of 10M cells.
- We can't infer the Q-value of new states from already explored states.
- Memory required to save and update that table would increase as the number of states increases.
- The time required to explore each state to create the required Q-table would be unrealistic.
- Approximate these Q-values with machine learning models such as a neural network

DEEP Q-NETWORK

- Neural network to approximate the Q-value function.
- All the experience is stored by the user in memory .
- Next action is determined by the maximum output of the Q-network.
- Loss function here is mean squared error of the predicted Q-value and the target Q-value – Q*.
- $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \times \max_a Q(S_{t+1}, a) \right) - Q(S_t, A_t)]$

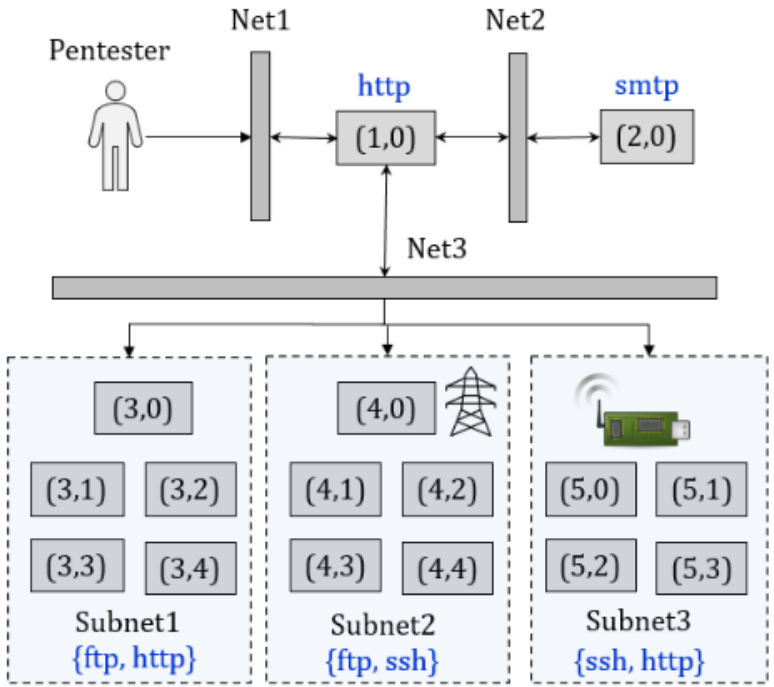


Multi-agent Reinforcement Learning (MARL)

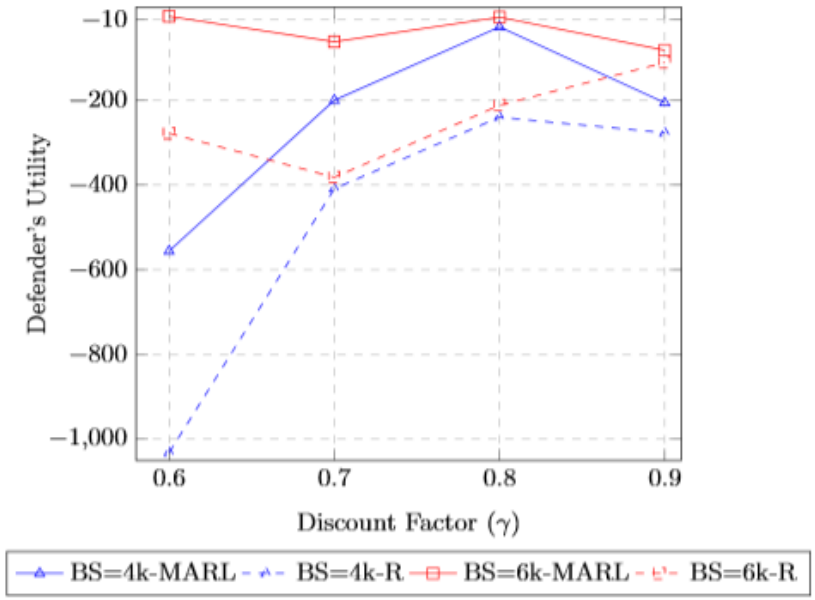
- We consider a sequential game where each agent in our model interacts with environment and other agent.
- CVSS score, attack/defense cost, and impact of MTD on security policies are considered.
- We consider MARL for modeling fully competitive game between attacker and defender.

Empirical Evaluation

- Simulated enterprise network representing Industrial Control System (ICS) – Subnet2, and IoT Network – Subnet3.
- Attack vectors and CVSS score based on scanning of services – http, ftp, ssh, smtp.
- MTD countermeasures Shuffle, IP Mutation used in defense.



KEY OBSERVATIONS



- MARL-4k and MARL-6k show reward from MARL with batch sizes (BS): 4k, 6k.
- Defender obtains highest reward using reinforcement learning strategy.
- Utility obtained for BS = 6k is higher compared to BS = 4k.
- Utility value increases with discount factor γ upto a certain value, i.e., $\gamma = 0.8$.

CONCLUSION

- MTD as a two-player zero-sum Markov Game.
- We represented adaptive-adversary using Multi-agent reinforcement learning (MARL).
- Introduced domain specific reward model which considers CVSS score, and attack/defense costs.
- Defender obtains higher reward by being adaptive (MARL) strategy against attacker.

Questions??