UNIVERSITY of HOUSTON

ADVERSARIAL REINFORCEMENT LEARNING BASED ADAPTIVE MOVING TARGET DEFENSE

TAHA EGHTESAD*, YEVGENIY VOROBEYCHIK†, ARON LASZKA*

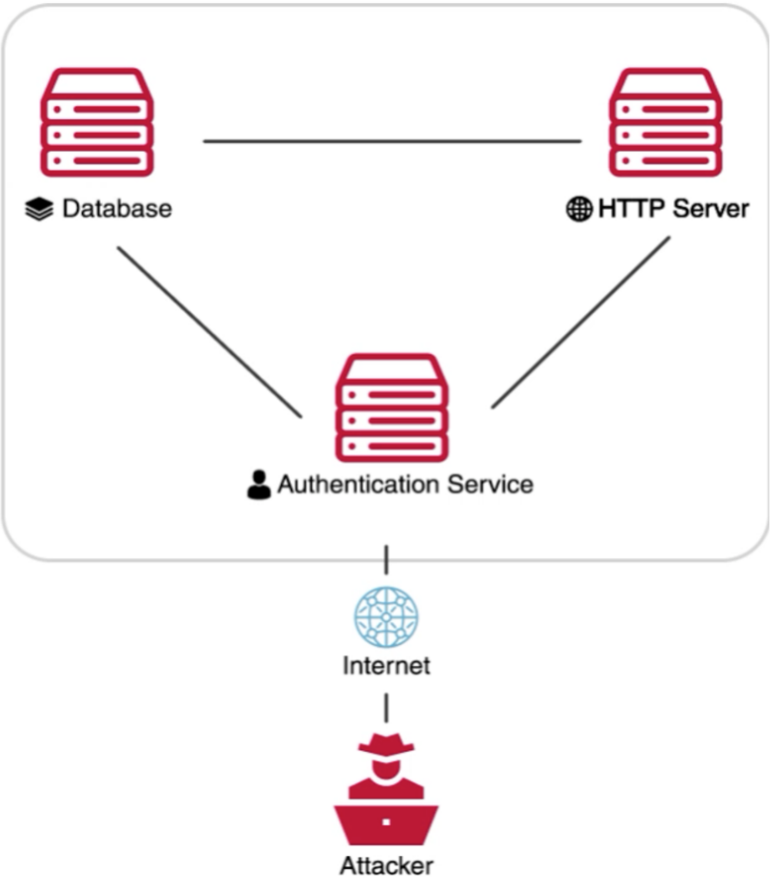
* UNIVERSITY OF HOUSTON

† WASHINGTON UNIVERSITY IN ST. LOUIS

11TH CONFERENCE ON DECISION AND GAME THEORY FOR SECURITY, GAMESEC'20

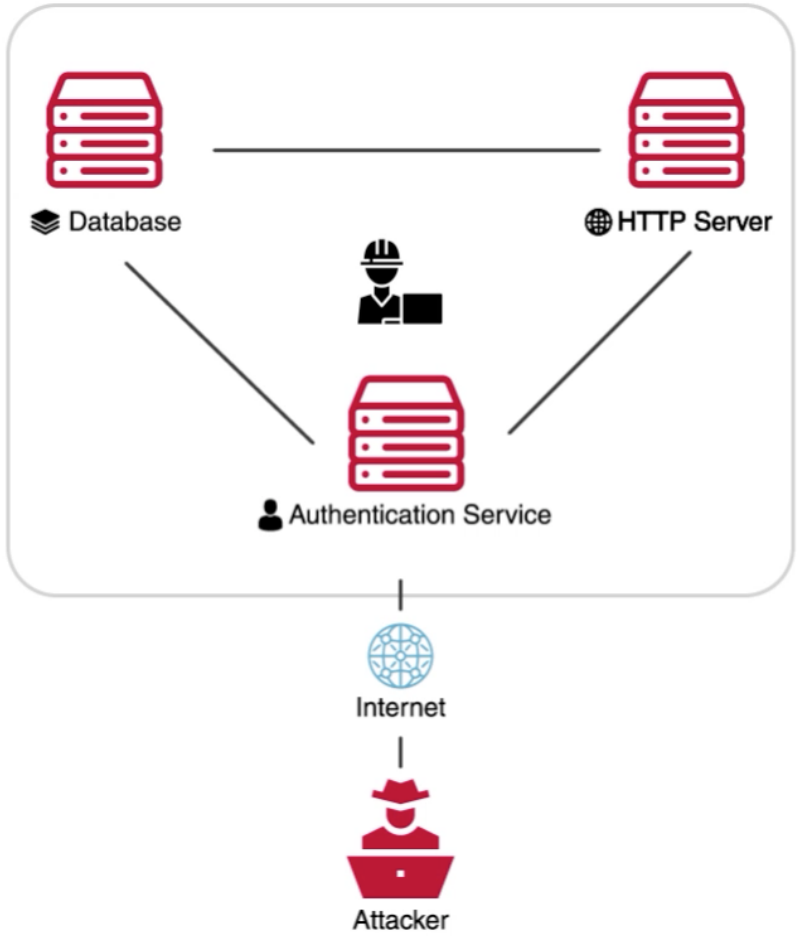
MOVING TARGET DEFENSE (I)

- Traditional Approach to security:
  - Hardening systems by finding and eliminating vulnerabilities.
    - They will not provide perfect security.
    - The attacker can use design flaws and implementation mistakes.
- Moving Target Defense:
  - A Proactive defense
    - Changing the configuration of assets randomly
      - For example: IP addresses, software deployments
    - Increases the uncertainty of the attacks.
      - Putting the adversary in an infinite loop of exploration



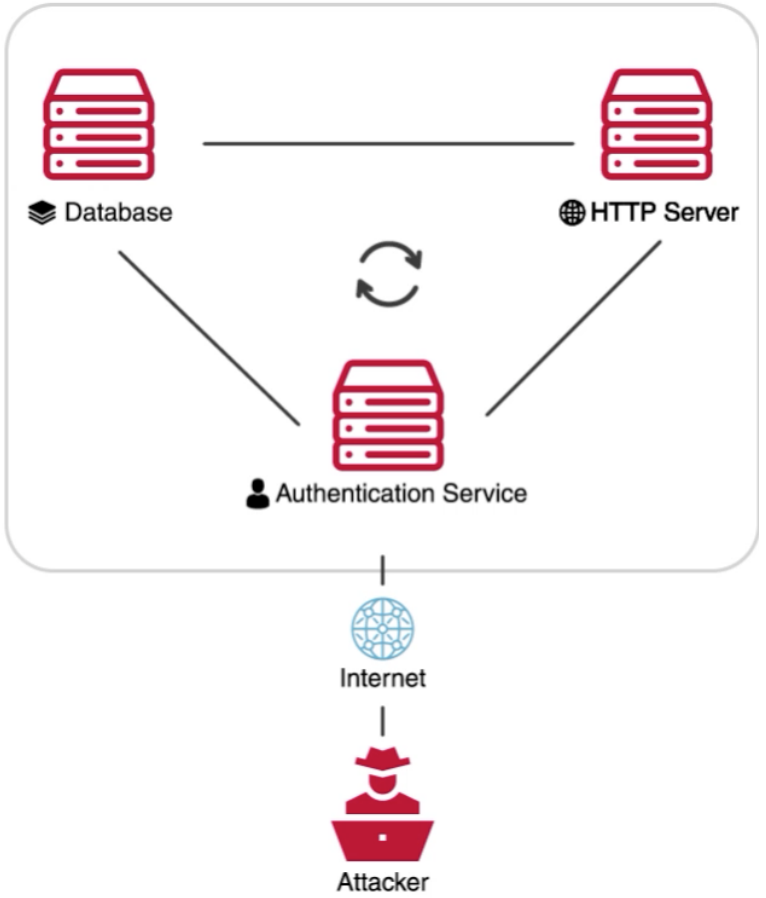
MOVING TARGET DEFENSE (II)

- MTD configurations should be deployed continuously.
- Currently, sysadmins ***manually*** select MTD configurations to be deployed.
  - Based on their experience.
    - Time consuming
      - Constraint on deployment locations.
      - Physical connectivity cannot be changed.
      - Resources are limited.
    - Trade off between security and efficiency
      - Most Secure: Total Randomization of configurations
      - Most Efficient: No change of configuration.



MOVING TARGET DEFENSE (III)

- We need an automated approach. Two key ingredients:
  - **Moving Target Defense Model**
    - Have been discussed in literature[1,2,3].
  - **Decision making algorithm**
    - Huge number of applicable MTD deployment combinations even with small in-control assets or MTD configurations.
    - Reinforcement Learning is one of the main approaches for decision making.
      - RL finds an optimal policy for a single agent in a static/dynamic environment



[1] Prakash, A., & Wellman, M. P. (2015, October). Empirical game theoretic analysis for moving target defense. In *Proceedings of the second ACM workshop on moving target defense* (pp. 57-65).
[2] Lei, C., Ma, D. H., & Zhang, H. Q. (2017). Optimal strategy selection for moving target defense based on Markov game. IEEE Access, 5, 156-169.
[3] Li, H., & Zheng, Z. (2019, November). Optimal timing of moving target defense: A Stackelberg game model. In MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM) (pp. 1-6). IEEE.

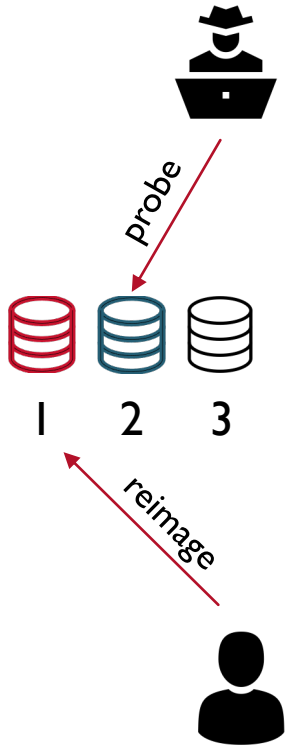
OVERVIEW OF PRESENTATION

1. Present a MTD model.[1]
2. Formulate the solution of the game based on Nash equilibria.
3. Propose an approach for solving the game using the Double Oracle (DO) algorithm.
4. Show how Reinforcement Learning (RL) can be used as a Best-Response (BR) Oracle.
5. Propose a framework for solving the MTD game.
6. Show our approach is computationally feasible, stable, and viable.

[1] Prakash, A., & Wellman, M. P. (2015, October). Empirical game theoretic analysis for moving target defense. In Proceedings of the second ACM workshop on moving target defense (pp. 57-65).

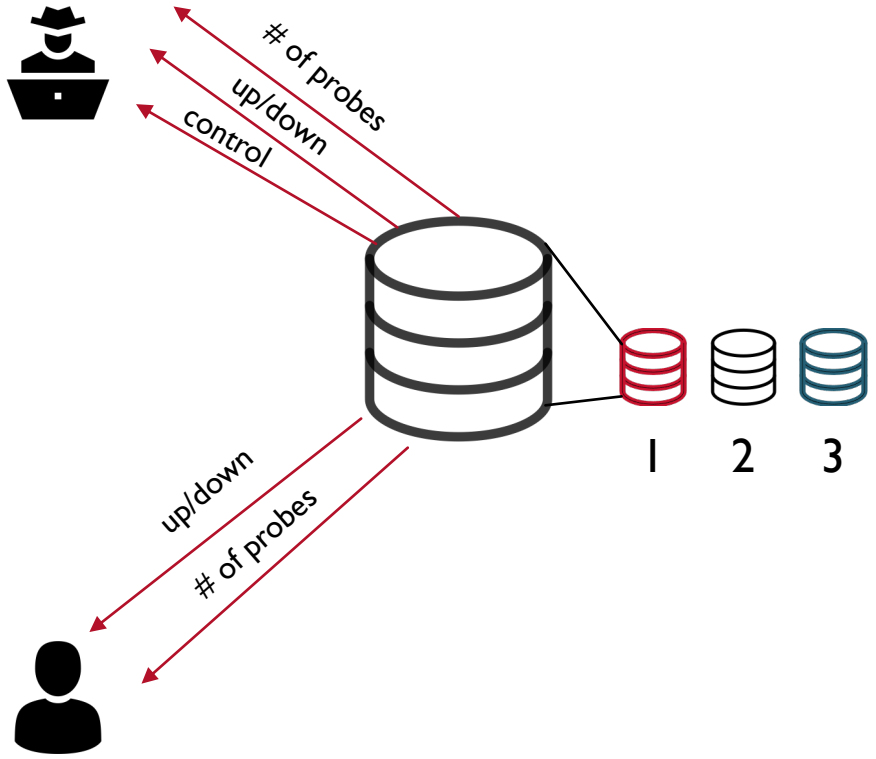
OVERVIEW OF MTD MODEL

- An Adversary (a) and a Defender (d) compete over set of M servers.
- In each discrete time steps, the Adversary ***probes*** a server.
  - The adversary compromises the server with some probability.
  - Or, increases the chance of compromising that server in future.
- The Defender ***reimages*** a server.
  - Takes the server down for fixed time steps.
  - Resets the adversary's progress on that server.
  - The Defender takes back the control of the server.
- Each player is rewarded based on the portion of servers that are in control or down.
- State of each server at each time step is presented as a tuple: $s_i^\tau = \langle \rho, \chi, \upsilon \rangle$.
  - ρ: Number of Probes, χ: control, υ: up/time to up



OBSERVATIONS

- **The Adversary:**
  - Estimates the number of probes on servers.
  - Learns whether a server is up or down by probing it.
  - Knows who controls which servers.
    - Knows when a compromised server is reimaged.
    - Does not know when a server that was not in-control is reimaged.
- **The Defender:**
  - Knows which servers are down.
  - Only observes a probe on a server with probability $1 - \nu$.
  - Does not know who controls which servers.



REWARDS

- Each player p gains reward based on a family of utility functions in form of sigmoid:
  - For simplicity:

$$r^a(n_c^a, n_d) = \frac{1}{e^{-0.5 \cdot (\frac{n_c^a + n_d}{M} - 0.5)}} - C_A$$

$$r^d(n_c^d) = \frac{1}{e^{-0.5 \cdot (\frac{n_c^d}{M} - 0.5)}}$$

- Implicit Defender reimage cost:
  - Not gaining reward from servers that are down
- Explicit Cost of Attack (C_A) for Adversary's probe

	Utility Environment	w^a	w^d
0	Control/Availability	1	1
1	Control/Confidentiality	1	0
2	Disrupt/Availability	0	1
3	Disrupt/Confidentiality	0	0

PURE STRATEGY

- A ***strategy*** is a policy function which given the current observation from the environment, produces an action to be taken by the agent:

$$\pi(o_\tau) \mapsto a_\tau$$

- A ***pure strategy*** is a deterministic policy function.
- Let Π^p denote the pure strategy sets of player p.
- The ***utility payoff*** of player p's pure strategy (π^p) versus the opponent's ($\bar{p}$) pure strategy ($\pi^{\bar{p}}$) can be calculated as:

$$U^p(\pi^p, \pi^{\bar{p}}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \cdot r_t^p \,\middle|\, \pi^p, \pi^{\bar{p}}\right]$$

- $\gamma \in [0, 1)$ is the discount factor which prioritizes immediate rewards over future rewards.

MIXED STRATEGY

- We need a mechanism to represent stochastic policies:
  - A ***mixed strategy*** for player p is a probability distribution σ^{p} over player p's pure strategies.
- Let Σ^{p} denote the mixed strategy sets of player p.
- The ***utility payoff*** of player p's mixed strategy (σ^{p}) versus the opponent's ($\bar{p}$) pure strategy ($\sigma^{\bar{p}}$) can be calculated as:

$$U^{p}(\sigma^{p}, \sigma^{\bar{p}}) = \Sigma_{\pi^{p} \in \Pi^{p}} \Sigma_{\pi^{\bar{p}} \in \Pi^{\bar{p}}} \sigma^{p}(\pi^{p}) \cdot \sigma^{\bar{p}}(\pi^{\bar{p}}) \cdot U^{p}(\pi^{p}, \pi^{\bar{p}})$$

PROBLEM FORMULATION

- The Adversary and the Defender are rational:
  - They always pick a strategy which maximizes their own utility.
- A ***best-response mixed strategy*** ($\sigma_*^p(\sigma^{\bar{p}})$) provides maximum utility for player p given that the opponent ($\bar{p}$) plays with strategy $\sigma^{\bar{p}}$:

$$\sigma_*^p(\sigma^{\bar{p}}) = \text{argmax}_{\sigma^p} U^p(\sigma^p, \sigma^{\bar{p}}) : \forall_{\sigma^p \in \Sigma^p}$$

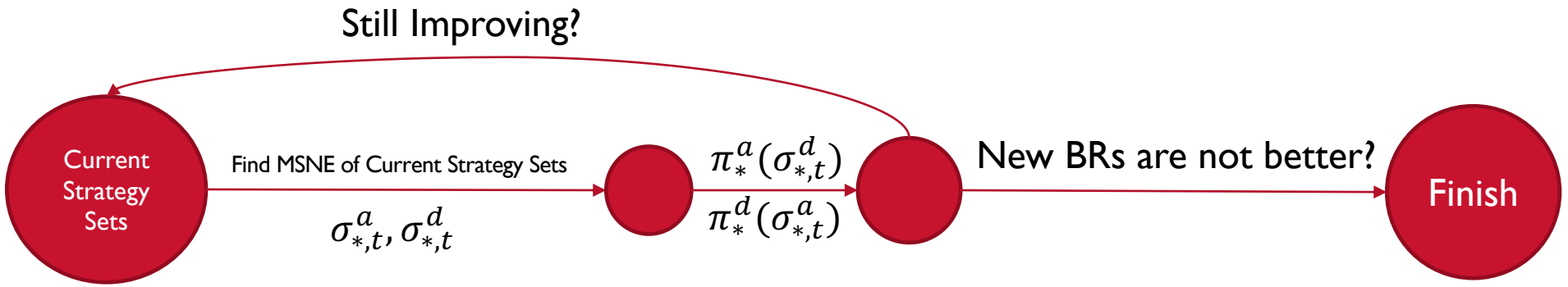
  - We need to optimize each player's strategy assuming that the opponent will always use a best-response.
  - This is equivalent of finding a ***Mixed Strategy Nash Equilibria.***
    - Find a pair for policies for the Adversary and Defender where neither player can increase its expected utility by unilaterally changing its strategy!

DOUBLE ORACLE

- The iterative Double Oracle algorithm solves a game given an arbitrary initial subset Π_0^p of each player p's strategy set ($\Pi_0^p \subset \Pi^p$):

$$\forall_{p\in\{a,d\}}: \Pi_{t+1}^p \leftarrow \Pi_t^p \cup \{\pi_*^p(\sigma_{*,t}^{\bar{p}})\}$$

  - In each iteration, the algorithm refers to a Best-Response Oracle:
    - Finding a best-response to the opponent's current dominant (MSNE) strategy.
    - Adding the best-response to the player's strategy set



DOUBLE ORACLE (CONT.)

- The best response (π_*^p) is calculated using a Best-Response (BR) Oracle.
- The algorithm is guaranteed to converge to *an MSNE* of the game.
  - The ***Equilibrium Selection Problem***:
    - The exact equilibria that the algorithm converges to depends on ***the initial strategy sets*** and the **output of the BR oracle**.
    - In experimental results, we show that this problem is not significant in our case.

REINFORCEMENT LEARNING AS BEST RESPONSE ORACLE

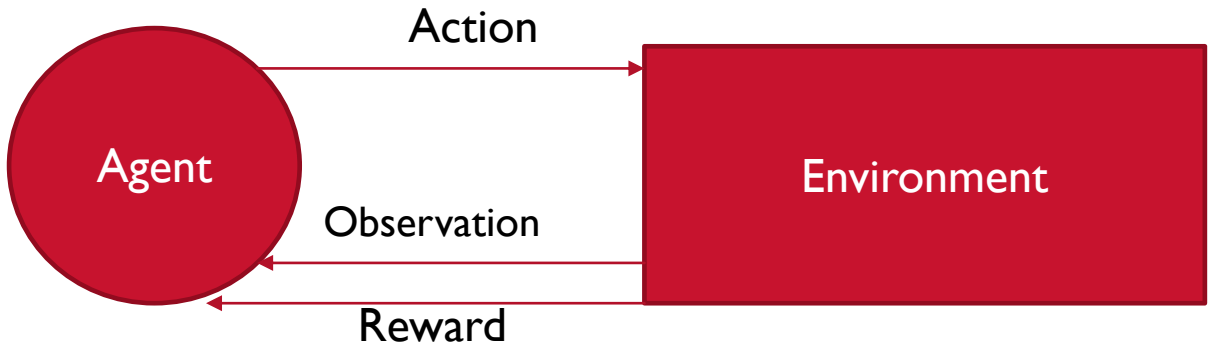
- Objective:
  - Find a policy function with trial and error.

$$\pi(o_\tau) \mapsto a_\tau$$

Which maximizes:

$$U_\tau^* = \mathbb{E}[\Sigma_{t=0}^{\infty} \gamma^t \cdot r_{t+\tau} | \pi]$$

- The training is done in steps:
  - observation, reward = env.step(a_t)
  - The agent stores these **experiences** in its ***experience replay buffer.***



$$e = \langle o_\tau, a_\tau, o_{\tau+1}, r_\tau \rangle$$

Q-LEARNING

- Uses a table to store Q values for each action/observation.
- The Q-value of a state-action estimates the expected future rewards of taking a given action at a given state.

$$Q(o_\tau, a_\tau) = U_\tau^*|_{\pi \leftarrow \text{argmax}_{\acute{a}} Q(o_\tau, \acute{a})}$$

  - Choose the action which results in highest expected utility of future rewards.
  - Update the Q-values based on ***experiences*** with ***Bellman Optimization Function:***

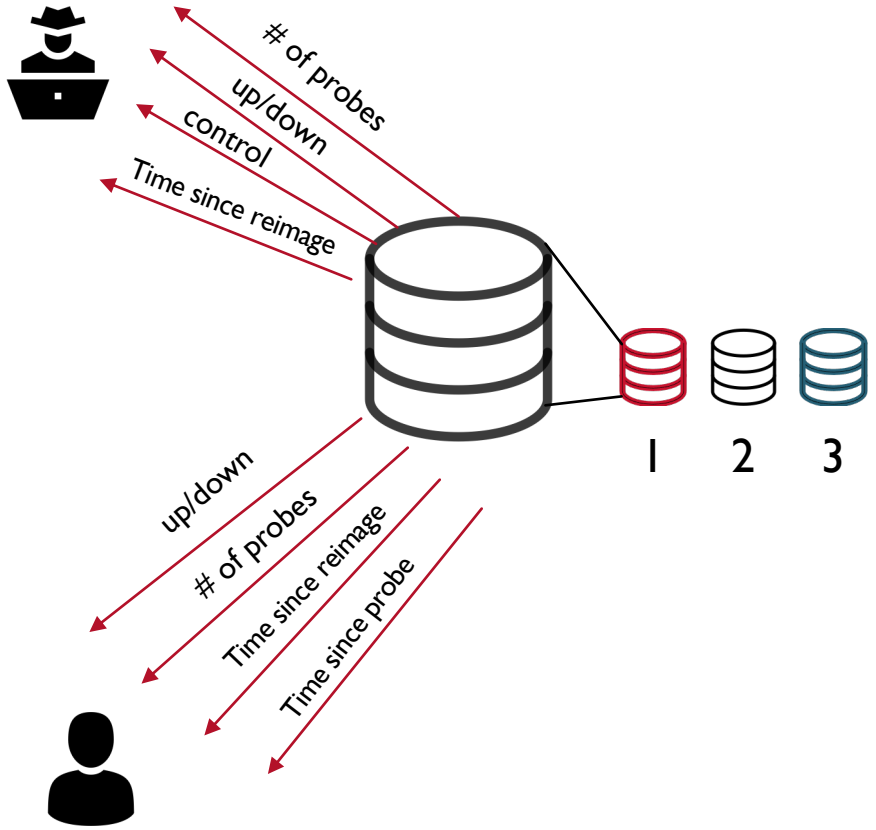
$$\mathrm{Q}(\mathrm{o}_\tau, \mathrm{a}_\tau) = \left(1 - \alpha_\mathrm{q}\right) \cdot \mathrm{Q}(\mathrm{o}_\tau, \mathrm{a}_\tau) + \alpha_\mathrm{q} \cdot (\mathrm{r}_\tau + \gamma \cdot \max_{\acute{\mathrm{a}}} \mathrm{Q}(\mathrm{o}_{\tau+1}, \mathrm{a}'))$$

IMPERFECT OBSERVATION (I)

- For both players, state is partially observable.
- Defender:
  - Does not know whether the servers are still in its control or not.
  - Example: The Defender has only observed a few probes on a server:
    - The server is safe?
    - The server is compromised?
- Adversary:
  - Does not know when uncompromised servers are reimaged.
  - Example: It had probed a server many time in the past:
    - Has it been reset?

IMPERFECT OBSERVATION (II)

- Policies consider a long history of preceding observations.
  - The size of the policy's effective State-Action space explodes!
- Compact Memory Representation:
  - Defender:
    - The amount of time since last reimaging (always known).
    - The amount of time since last probe.
  - Adversary:
    - The amount of time since last probe (always known).



COMPLEXITY OF MSNE COMPUTATION

- The MTD game is general-sum.
  - In some cases, it can be a constant sum.
- Problem of finding MSNE of given strategy sets in a general-sum game is PPAD-Complete.
  - MSNE computation in a game of non-trivial size is infeasible.
- ϵ-equilibrium:
  - Global Newton Method for finding the MSNE.

MODEL COMPLEXITY

- For storing a Defender policy's Q-values in a table, $(2T^3)^M \cdot M$ entries are required.
  - Policies for a game of non-trivial size will not fit in memory.
  - This many states-actions can not be explored even once.
- Solution is ***Deep-Q-Learning:***
  - Approximate the Q-values with neural networks.
  - Only hundreds/thousands of parameters needs to be stored.
    - Relation between states are generalized.
    - States are not required to be explored multiple times.

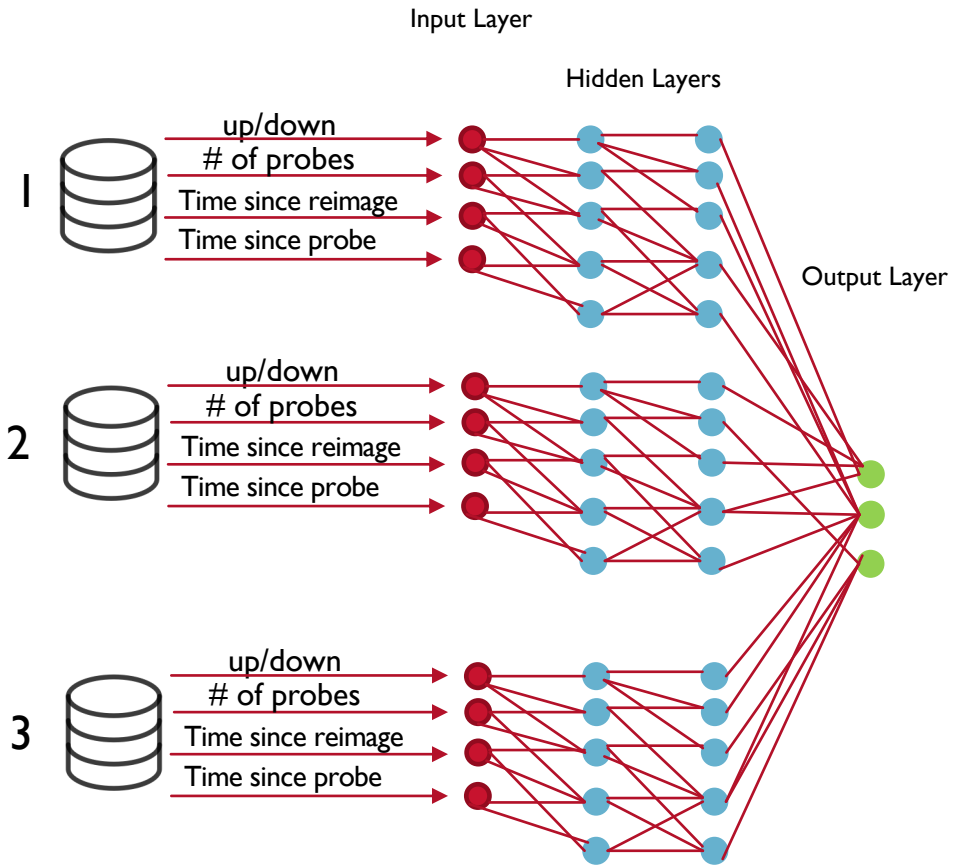
DEEP-Q-LEARNING

- A *Fully-Connected Feed-Forward Artificial Neural Network* approximates the Q-values.
  - For each state variable, one input neuron.
  - For each possible action, one output neuron.
  - The target Q-values are:

$$q_t = r_t + \gamma \cdot Q(o_{t+1}, \text{argmax}_{\acute{a}} Q(o_t, \acute{a}|\theta)|\theta)$$

  - Updating the network parameters can be done by taking an error function (e.g., MSE) and optimize it with *gradient descent* on a sample X of experiences.

$$L_\theta = \frac{1}{|X|} \Sigma_i^X \left(q_t - Q(o_t, a_t|\theta)\right)^2$$



PROPOSED FRAMEWORK

- Initialize strategy sets with No-OP Agents.
- While not converged:
  - Find MSNE of current Strategy Sets
  - Train an Adversary against MSNE of Defender.
  - Train a Defender against MSNE of the Adversary.
  - Assess newly found policies against previous strategies.

Algorithm 2: Adaptive Solver

Result: set of pure policies Π^a and Π^d

$\Pi^a \leftarrow$ attacker heuristics;

$\Pi^d \leftarrow$ defener heuristics;

while $U^p(\sigma^p, \sigma^{\bar{p}})$ *not converged* **do**

  $\sigma^a, \sigma^d \leftarrow$ solve_MSNE(Π^a, Π^d);

  $\theta \leftarrow$ random;

  $\pi^a_+ \leftarrow$ train($T \cdot N_e, env^a[\sigma^d], \theta$);

  $\Pi^a \leftarrow \Pi^a \cup \pi^a_+$;

  assess π^a_+;

  $\sigma^a, \sigma^d \leftarrow$ solve_MSNE(Π^a, Π^d);

  $\theta \leftarrow$ random;

  $\pi^d_+ \leftarrow$ train($T \cdot N_e, env^d[\sigma^a], \theta$);

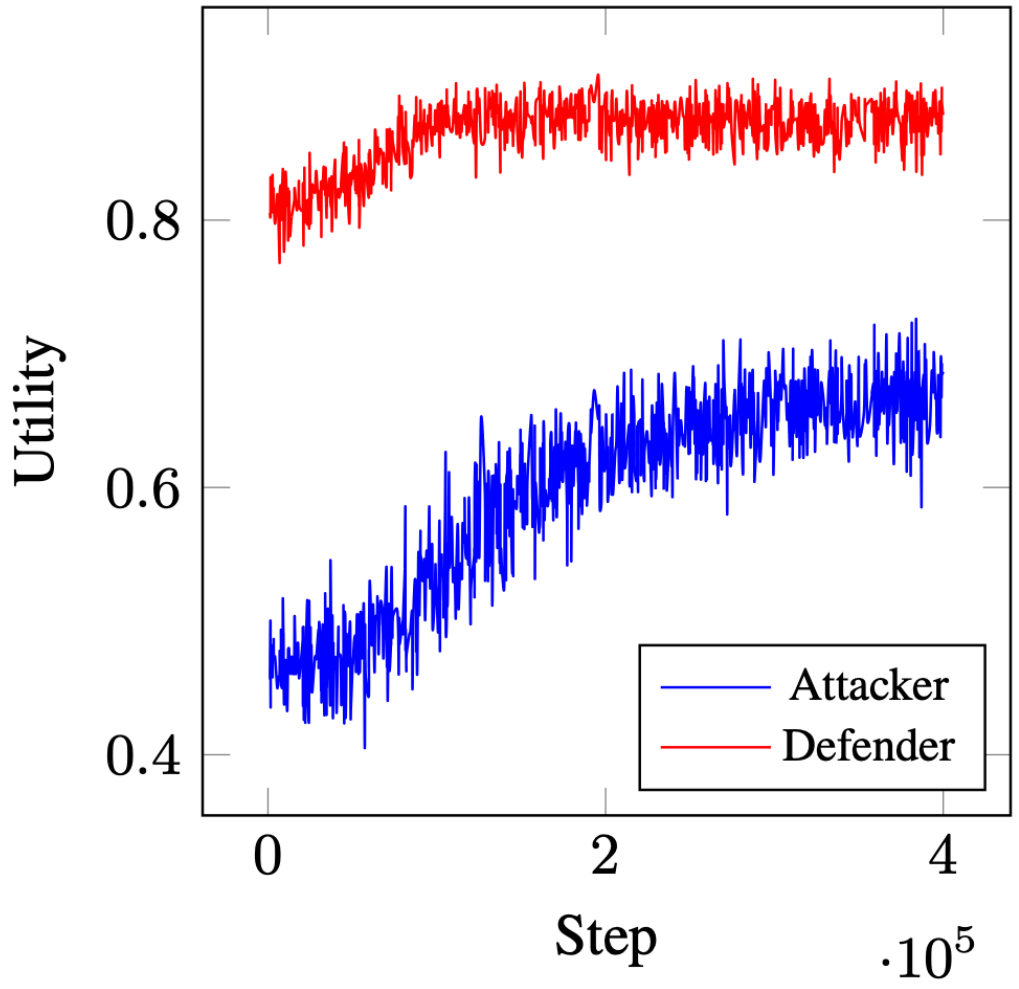
  $\Pi^d \leftarrow \Pi^d \cup \pi^d_+$;

  assess π^d_+;

end

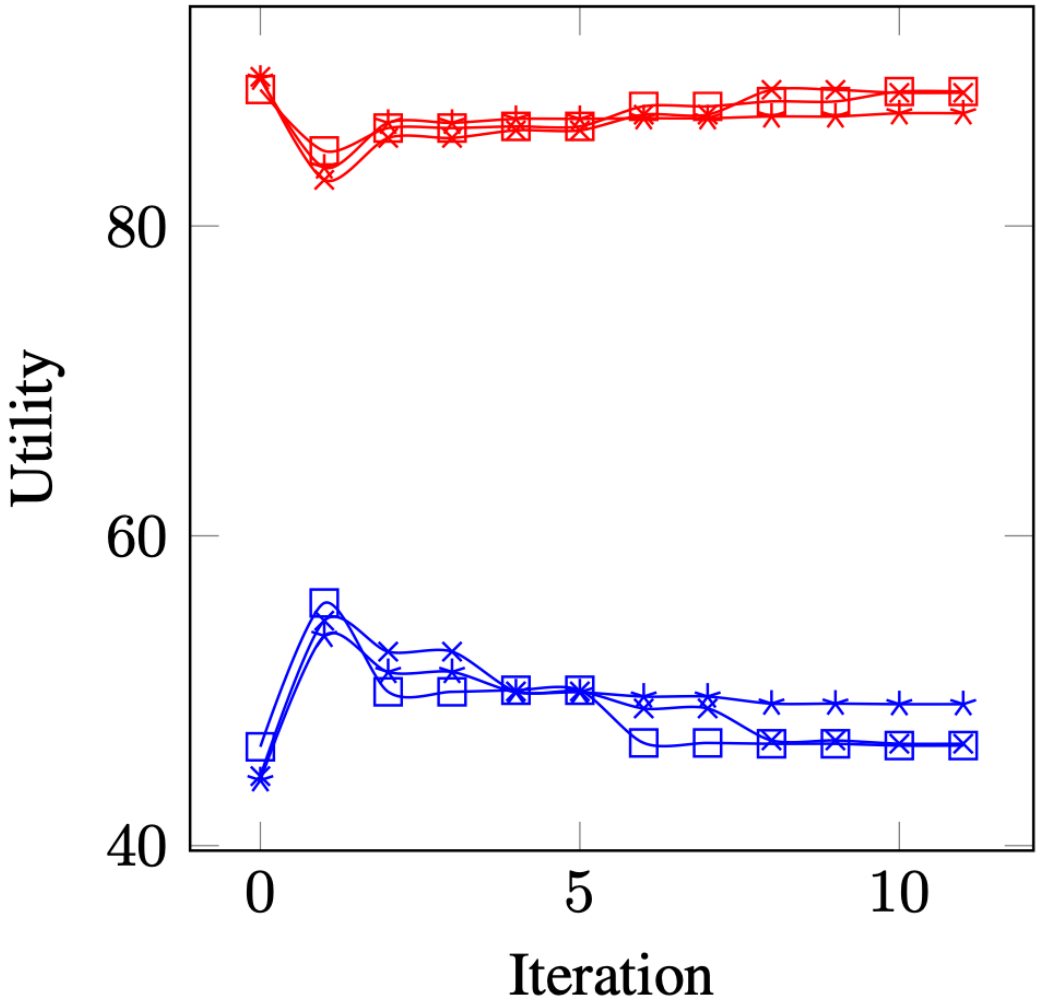
LEARNING CURVE

- Deep-Q-Learning works.
- The Adversary converges in $3.88 \cdot 10^5$ steps.
- The Defender converges in $1.10 \cdot 10^5$ steps.



DO CONVERGENCE - EFFICIENCY

- 3 DO Runs are extracted.
- Iteration 0 shows the payoff of heuristics.
- In all 3 cases, the DO algorithm converges in 8 iterations.
  - 6 hours of training.
  - DO with multiple approximations work!
  - Solution is computationally **Efficient**.



STRATEGY PAYOFFS

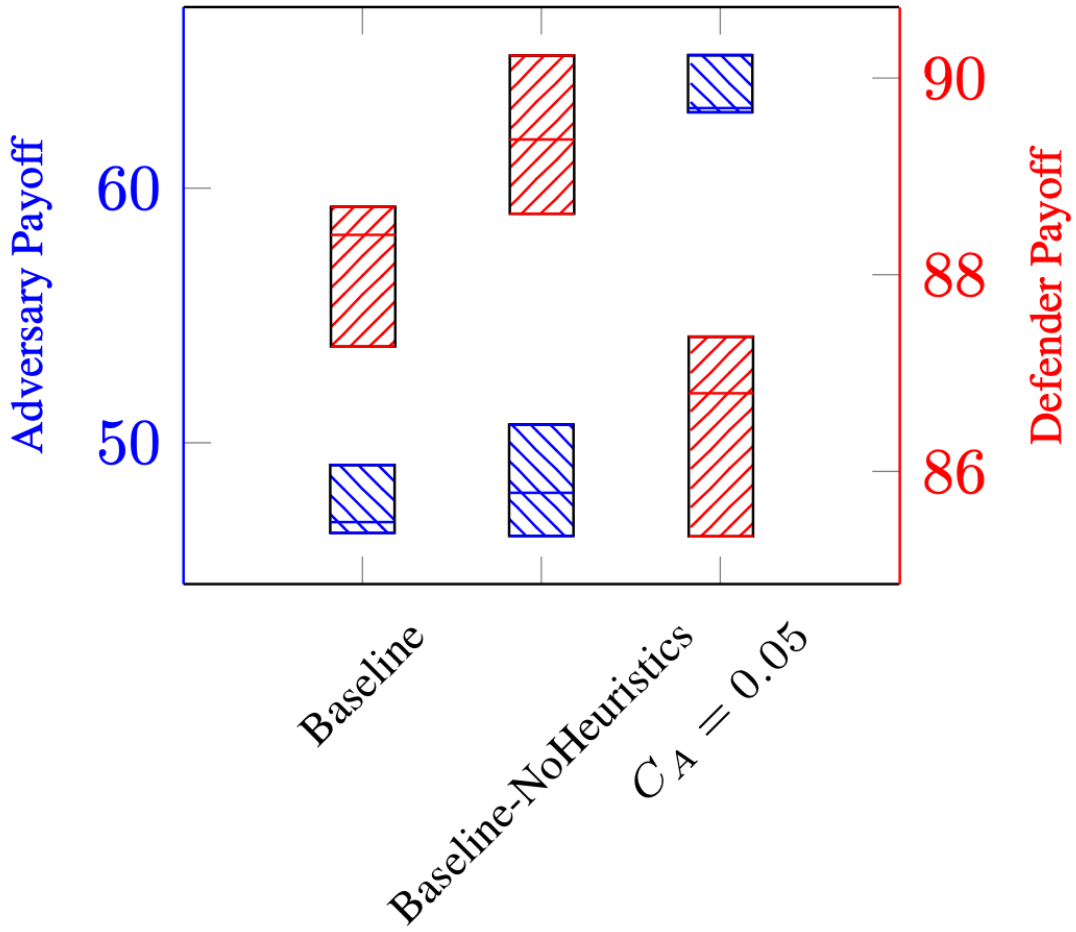
- Heuristics are extracted from the MTD model[1].
- Solution is **Viable**.
- Baseline: $M = 10, T = 1000, \gamma = 0.99$

Adversary \ Defender	No-OP	ControlThreshold	PCP	Uniform	MaxProbe	Mixed-Strategy DQL
No-OP	98.20 / 26.89	98.20 / 26.89	98.20 / 26.89	95.83 / 46.03	98.20 / 26.89	97.47 / 33.23
MaxProbe	47.69 / 78.66	49.62 / 75.67	93.01 / 36.58	67.12 / 64.56	86.82 / 41.99	87.84 / 45.87
Uniform	46.74 / 79.08	51.58 / 70.97	89.48 / 44.43	76.23 / 56.83	75.21 / 57.14	88.16 / 45.91
ControlThreshold	85.98 / 63.64	85.35 / 65.58	88.81 / 46.38	81.32 / 59.54	80.09 / 60.43	87.91 / 45.91
Mixed-Strategy DQL	72.29 / 62.78	82.45 / 58.31	91.32 / 45.76	87.10 / 55.31	91.32 / 44.57	92.38 / 45.23

[1] Prakash, A., & Wellman, M. P. (2015, October). Empirical game theoretic analysis for moving target defense. In Proceedings of the second ACM workshop on moving target defense (pp. 57-65).

EQUILIBRIA

- Equilibrium Selection Problem is not an issue in our case.
- The policies are resilient to under/over estimation.
- The Solution is **Stable**.



THANKS FOR YOUR ATTENTION.

ANY QUESTIONS?

Taha Eghtesad
teghtesad@uh.edu