Adaptive Learning for Moving Target defence: Enhancing Cybersecurity Strategies

Optimization Problems REIated to Network - OPEN CoDIT Workshop
11th International Conference on Control, Decision and Information Technologies

Mandar Datar, Yann Dujardin

Orange Research, France

July 15, 2025

Introduction

- **Cybersecurity** aims to prevent cyberattacks such as ransomware, data breaches, and system damage.
- With increasing internet usage and network traffic, **automated defence mechanisms** are more essential than ever.
- Traditional defences follow a *detection and response* approach, but systems remain vulnerable.
- A key challenge: **information asymmetry** — attackers often know more than defenders.

Moving Target Defense (MTD)

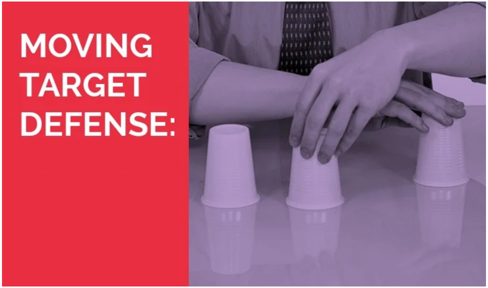


Figure: Changing the attack surface

Definition

Moving Target Defense (MTD) is a cybersecurity strategy that aims to prevent cyberattacks by frequently changing the attack surface or system configuration, making it more difficult for attackers to identify and exploit vulnerabilities.

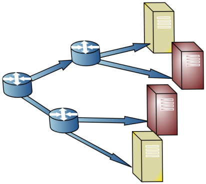
Security measures Vs Potential performance

- However, MTD introduces a tradeoff:
  - Frequent reconfiguration $\Rightarrow$ performance degradation.
  - Infrequent reconfiguration $\Rightarrow$ higher security risk.

- For instance, degradation in system performance and the dissatisfaction of customers due to delays etc.

- **Goal:** optimize reconfiguration frequency to balance *security* and *performance*.

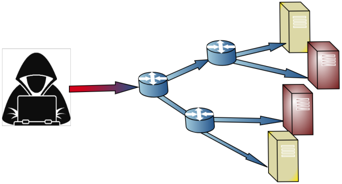
System model



We consider a single system, which can represent, in general, a critical cyber system

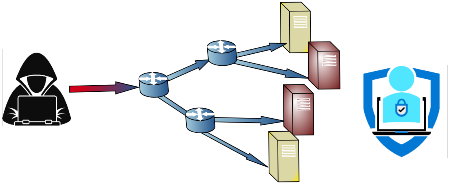
- MTD has been applied to secure system
- It involves techniques like system reconfiguration, code and data diversification, and network topology changes

Attacker



- Attacker attempts to gain control over a system through a series of probing actions.
- With each probe, he either can succeed in gaining control of the system or it increases the attacker's chances of gaining control in subsequent attempts.

Defender



- Defender periodically re-image the system
- Any progress made by the attackers in infiltrating the system completely wiped out during the reimage process.

Game Model

- We model the interaction between attacker and defender as a **partially observable stochastic game**
- Two players, Defender and Attacker, compete to gain control over a system or a security-sensitive resource.
- Nash equilibrium is a strategy profile where no player can do better by unilaterally changing their strategy.

Actions and Rewards

- Defender can reimage the system which will cost him C_D
- Attacker can probe the system, probing the system takes control of it with probability
$$1 - e^{-\alpha(\rho+1)} \tag{1}$$
and for each probe, it costs C_A for the attacker.
- A player gets a reward of 1 unit if it controls the system at that time period.

State and Action Space

We define the states of the system as follows:

$$\mathcal{S} = \{0, 1\}$$

0 defender controls the system

1 attacker controls the system

We define action sets for defender and attacker as :

- $\mathcal{A}_D = \{0, 1\}$, where 0:=reimage, 1:=continue
- $\mathcal{A}_A = \{0, 1\}$, where 0:=probe, 1:=not probe

Transition Probability

$\mathcal{T}(j, i, d, a)$ denotes the probability of transition from state i to state j when the defender and attacker take the actions d and a respectively. The transition probabilities can be summarized as follows:

$$\mathcal{T}(0,0,1,0) = e^{-\alpha} \quad (2)$$

$$\mathcal{T}(1,0,1,0) = 1 - e^{-\alpha} \quad (3)$$

$$\mathcal{T}(0,\cdot,0,\cdot) = 1 \quad (4)$$

$$\mathcal{T}(0,0,1,1) = 1 \quad (5)$$

$$\mathcal{T}(1,1,1,.) = 1 \quad (6)$$

Observations

- The defender does not know whether the attacker has compromised the system or not.
- The defender can observe the each probe with probability $1 - \nu$ and with probability ν probe is undetected.
- The attacker cannot observe the re-imaging of the uncompromised system without probing it

Belief about the game state

- Depending on observations players form the belief about the game state

$$b_{D,t} = \mathbb{P}\left[s_t | h_t^1\right] \tag{7}$$
$$b_{A,t} = \mathbb{P}\left[s_t | h_t^2\right] \tag{8}$$

The belief update can be expressed as follows:

$$b_{t+1} = T_{\pi_A}\left(o_{t+1}, b_t, d_t\right) \tag{9}$$

$$b'(s') = \frac{P(o|d, s') \cdot \sum_s T(s, d, s') \cdot b(s)}{\sum_{o'} P(o'|d, s')} \tag{10}$$

Where:

$b'(s')$: Updated belief state after taking action a and observing o in state s'

$T(s, d, s')$: Transition probability of transitioning from state s to state s' when takir

$b(s)$: Initial belief state before the update

$P(o|a, s')$: Probability of observing o when taking action a in state s'

Objective Functions(Maximizing Discounted Rewards)

$$J_D(\pi_D, \pi_A) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_D(s_t, d_t, s_{t+1}) \middle| \pi_D, \pi_D\right] \tag{11}$$

$$J_A(\pi_D, \pi_A) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_A(s_t, a_t, s_{t+1}) \middle| \pi_D, \pi_D\right] \tag{12}$$

- π_D and π_A represent the policies for Defender and Attacker, respectively.
- $R_D(s_t, a_t, s_{t+1})$ is the reward for Defender at time step t.
- $R_A(s_t, a_t, s_{t+1})$ is the reward for Attacker at time step t.
- γ is the discount factor.
- s_t represents the state at time step t.
- d_t and a_t represent the actions chosen by Defender and Attacker at time step t.
- The expectations are taken with respect to the joint policies π_D and π_A.

Nash Equilibrium

We assume that both the players want to maximize their reward. A policy pair (π_D^*, π_A^*) is said to be Nash equilibrium if no player can benefit by deviating unilaterally.

$$J_D(\pi_D, \pi_A) = \mathbb{E}_{(\pi_D,\pi_A)}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_D(s_t, a_t, d_t)\right], \tag{13}$$

$$J_A(\pi_D, \pi_A) = \mathbb{E}_{(\pi_D,\pi_A)}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_A(s_t, a_t, d_t)\right]. \tag{14}$$

Bellman Equation for POMDP

Since a POMDP is a continuous-state MDP with state space being the unit simplex, we can straightforwardly write down the dynamic programming equation for the optimal policy as for continuous-state MDPs.

$$V_{\pi_A}(b) = \mathrm{Max}_{d \in \mathcal{A}_D} \mathbb{E}_{\pi_A} \left[R_D(b,d) + \sum_{o \in \mathcal{O}} V_{\pi_A}(T(o,b,d))\sigma(o,b,d) \right] \tag{15}$$

For the attacker,

$$V_{\pi_D}(s,b) = \mathrm{Max}_{d \in \mathcal{A}_A} \mathbb{E}_{\pi_D} \left[R_A(s,a) + \sum_{s' \in \mathcal{S}} V_{\pi_D}(s')P(s',s,a) \right] \tag{16}$$

Structural results for Policy

Theorem

Given an attacker policy $\pi_A \in \Pi_D$, the defender's value function $V_{\pi_A}(b)$ is decreasing in the belief state b. Moreover, the defender's optimal policy follows a threshold structure, i.e., it is decreasing in b.

Theorem

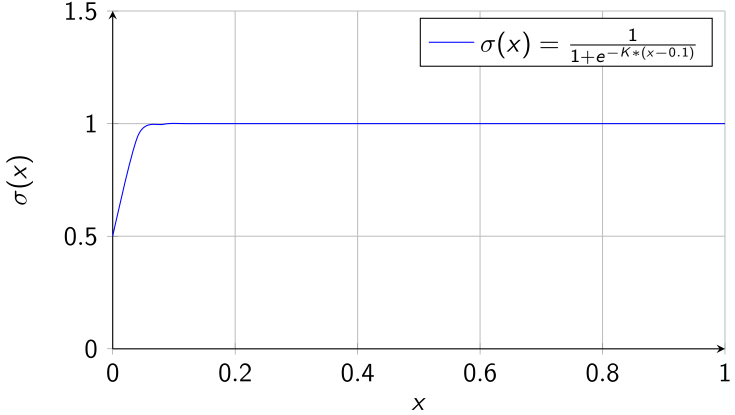
Given a fixed defender policy $\pi_D \in \Pi_D$, the attacker's value function $V_{\pi_D}(s, b)$ is increasing in the state s. Furthermore, the attacker's optimal policy exhibits a threshold structure: it is increasing in s. Moreover, since the defender's policy π_D is decreasing in the belief b (as established in the previous theorem), the attacker's optimal policy is increasing in b.

The parameterized threshold policy

Let b be the current belief in the state and let θ be a parameter. The parameterized threshold policy using a sigmoid function can be defined as follows:

$$\pi(a|b,\theta) = \frac{1}{1+e^{-K(b-\theta)}} \tag{17}$$

Threshold policy



Policy Gradient Theorem

Theorem (Policy Gradient Theorem)

The policy gradient is given by:

$$\nabla J(\theta) \propto \mathbb{E}_\pi \left[\nabla \log \pi(a|b) Q(b, a)\right]$$

Where:

- $\nabla J(\theta)$ *is the gradient of the expected return with respect to the policy parameters* θ*.*
- $\pi(a|b)$ *is the policy.*
- $Q(b, a)$ *is the action-value function.*

Policy Gradient-Based Fictitious Play

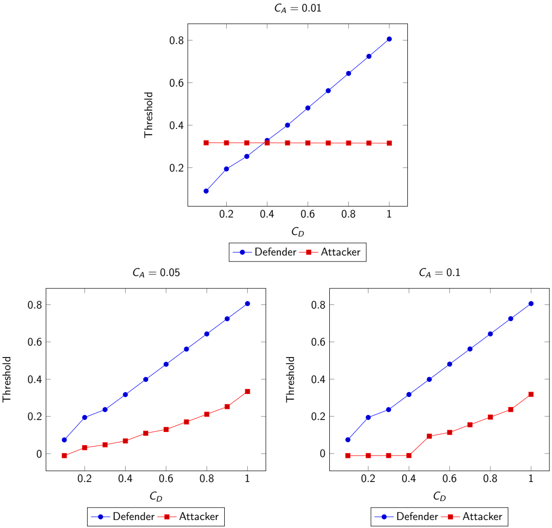
Algorithm Policy Gradient-Based Fictitious Play for Stochastic Game

1: Initialize the threshold policy for each player: Defender and Attacker.
2: **repeat**
3:     **for each** Player $s \in \{\text{Attacker, Defender}\}$ **do**
4:         Consider the policy of the opponent as fixed.
5:         Collect trajectories using policies π_{θ_s}.
6:         Compute the returns $R_s(\tau)$.
7:         Compute the policy gradients $\nabla_{\theta_s} J_s(\theta_s, \theta_{-s})$.
8:         Update the policy parameters:

$$\theta_s \leftarrow \theta_s + \alpha_s \nabla_{\theta_s} J_s(\theta_s, \theta_{-s})$$

9:     **end for**
10:     Repeat steps 2 and 3 until convergence.
11: **until** convergence

Numerical Experiments



Thank You

Thank You