Learning Equilibria in Stochastic Information Flow Tracking Games with Partial Knowledge

Shruti Misra*, Shana Moothedath*, Hossein Hosseini*, Joey Allen†, Linda Bushnell*, Wenke Lee†, and Radha Poovendran*

*Network Security Lab, Department of Electrical and Computer Engineering, University of Washington, Seattle, WA, USA
†School of Computer Science, College of Computing, the Georgia Institute of Technology, Atlanta, GA, USA

Advanced Persistent Threats

An advanced persistent threat (APT) is a prolonged and targeted cyberattack in which an intruder gains access to a network and remains undetected for a period of time.

The intention of an APT attack is usually to monitor network activity and steal data rather than to cause damage to the network or organization.

2013 Target Corporation Data Breach

40 million credit and debit card numbers and 70 million records of personal information were stolen

- **September 2013** — Phishing attacks against Fazio Mechanical Services
- **15 November 2013** — Accessed Target's network and tested malware on Point of Service machines
- **27 November 2013** — Began collection of credit card data from Point of Service machines
- **2 December 2013** — Moved data out of Target's network

Shu, X., Tian, K., Ciambrone, A. and Yao, D., 2017. Breaking the target: An analysis of target data breach and lessons learned. *arXiv preprint arXiv:1701.04940*.

Stages of Advanced Persistent Threats (APTs)

40 million credit and debit card numbers and 70 million records of personal information were stolen

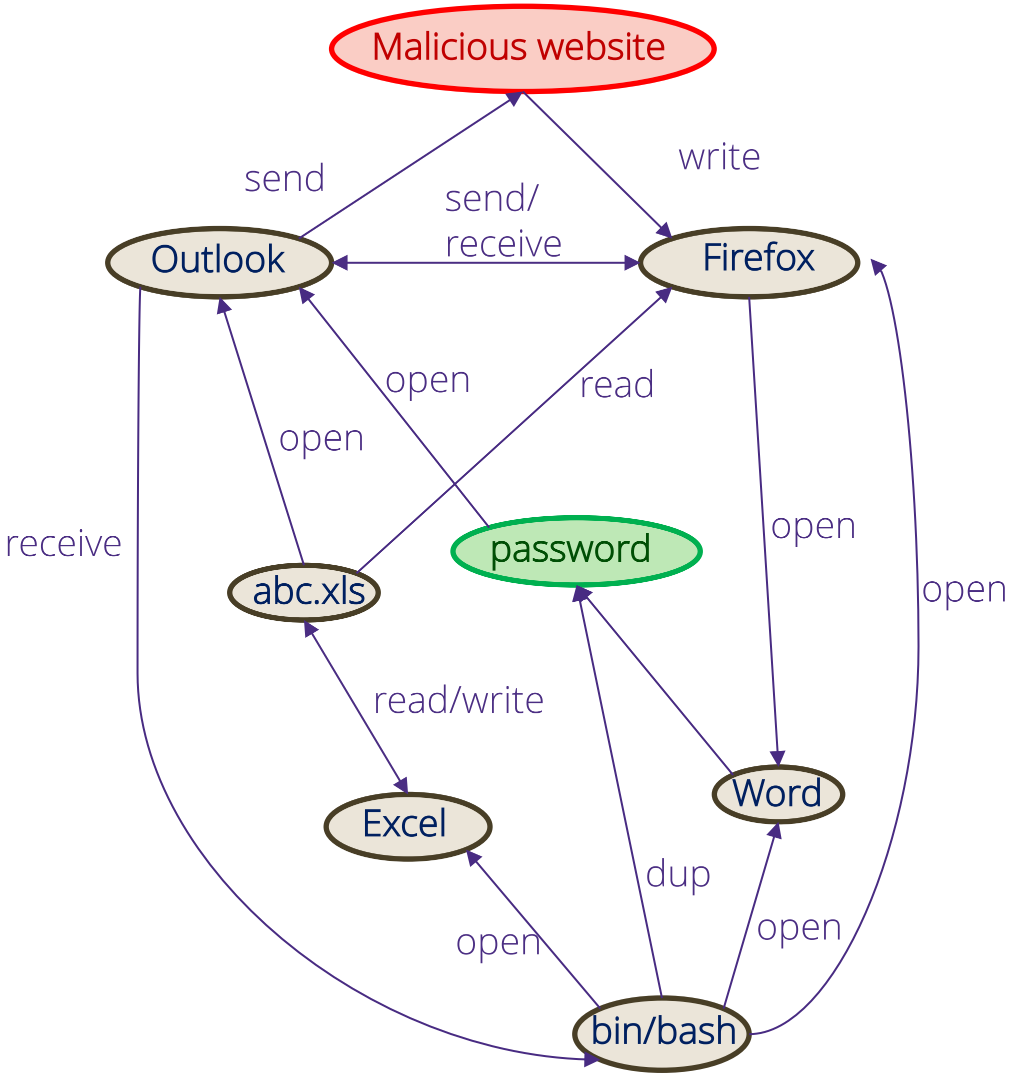
Initial Infection	Phishing attacks against Fazio Mechanical Services
Command and Control	Accessed Target's network and tested malware on Point of Service machines
Lateral Expansion	Began collection of credit card data from Point of Service machines
Data Exfiltration	Moved data out of Target's network

Shu, X., Tian, K., Ciambrone, A. and Yao, D., 2017. Breaking the target: An analysis of target data breach and lessons learned. *arXiv preprint arXiv:1701.04940*.

Information Flow Graphs (IFGs)

Adversary interact with system components (e.g. processes, files) using system calls (e.g. read, write, open)

Information flows (data- and control-flow commands) abstract the interaction of the adversary with various system components.

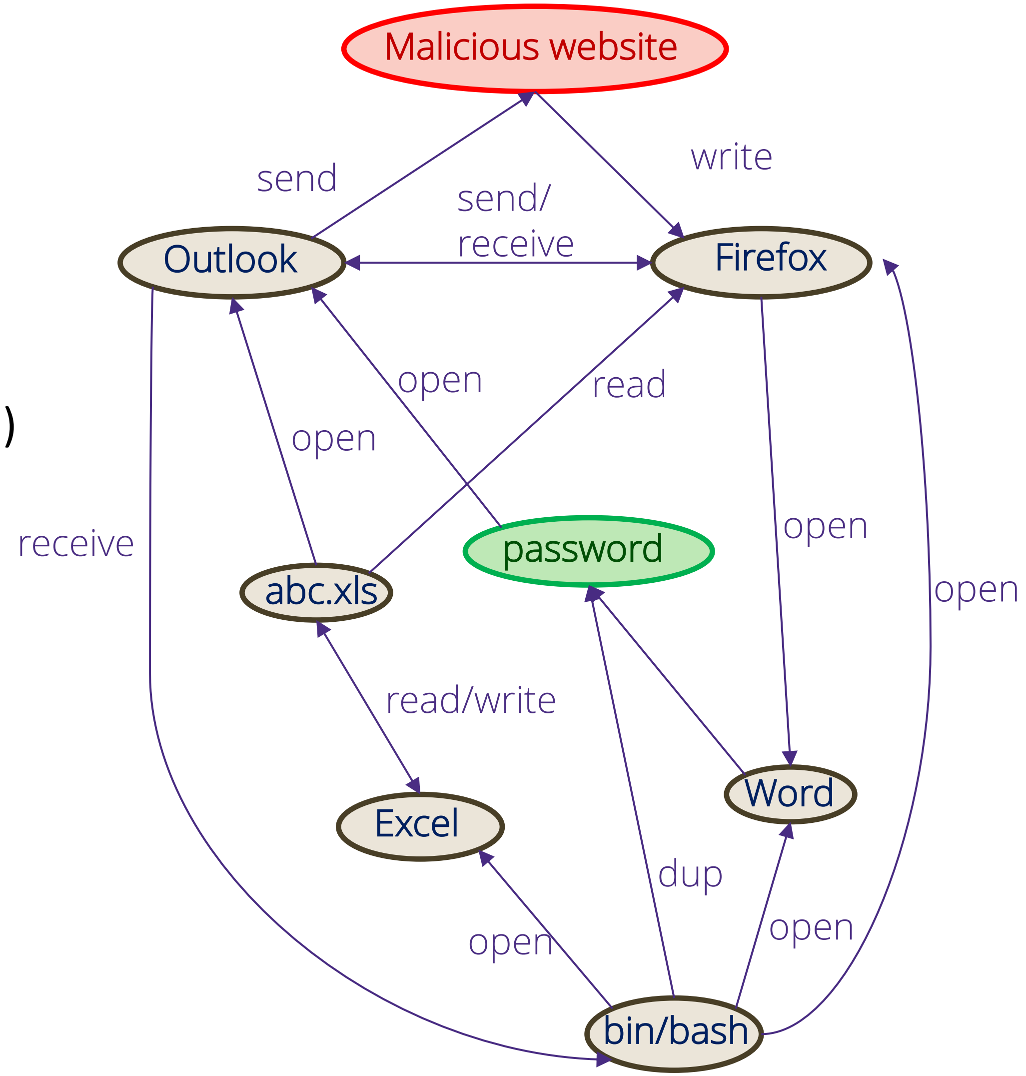


Y. Ji, S. Lee, E. Downing, W. Wang, M. Fazzini, T. Kim, A. Orso, and W. Lee. "RAIN: Refinable Attack Investigation with On-demand Inter-Process Information Flow Tracking." In *ACM SIGSAC CCS*, 2017.

Information Flow Graphs (IFGs)

Graphical representation of system log data

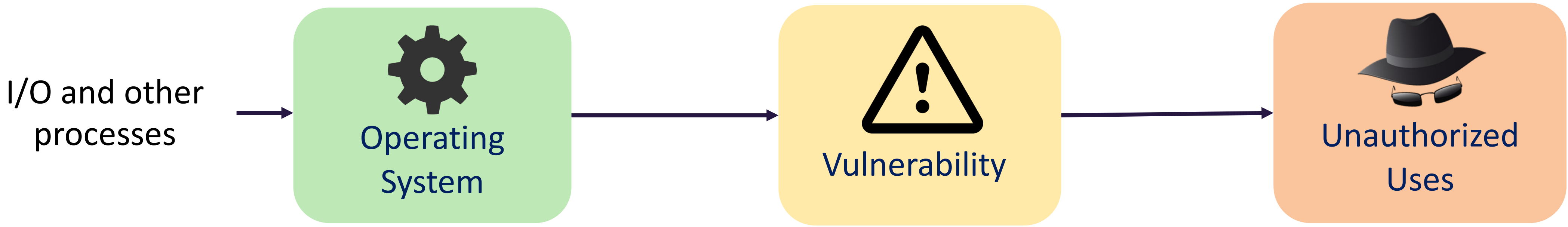
- Nodes: Subjects and Objects
  - Subjects: Processes (an instance of a computer program)
  - Objects: e.g. files/memory/network sockets
- Edges: probability of transferring information flows between processes
  - System calls: e.g. read, write, open, send



Y. Ji, S. Lee, E. Downing, W. Wang, M. Fazzini, T. Kim, A. Orso, and W. Lee. "RAIN: Refinable Attack Investigation with On-demand Inter-Process Information Flow Tracking." In *ACM SIGSAC CCS*, 2017.

Dynamic Information Flow Tracking (DIFT)

DIFT is a flow tracking-based mechanism that is widely used to detect APTs.



Tag suspicious data channels

Track propagation of tagged inputs based on tag propagation policy (data- and control-flow)

Trap attacks; check the authenticity of tagged flows and restrict spurious values

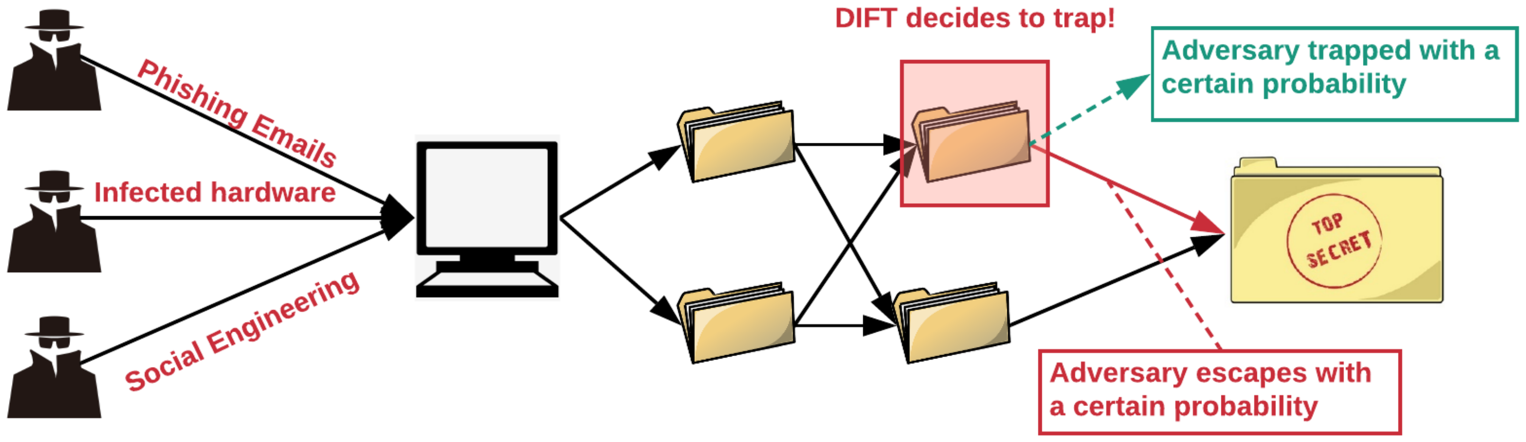
Suh, G. E., Lee, J. W., Zhang, D., & Devadas, S. (2004, October). Secure program execution via dynamic information flow tracking. *In ACM Sigplan Notices,*ACM.

Dynamic Information Flow Tracking (DIFT)

Implementation and operation of DIFT, however, introduce memory and performance overhead on the system as it involves tracking and analyzing a large number of benign flows [3]. Thus an optimal selection of processes in the system to perform security analysis is critical for effective and resource efficient detection.

Suh, G. E., Lee, J. W., Zhang, D., & Devadas, S. (2004, October). Secure program execution via dynamic information flow tracking. *In ACM Sigplan Notices,*ACM

Problem Formulation



Aim

Model a cost-effective DIFT-based defense mechanism against APTs that:

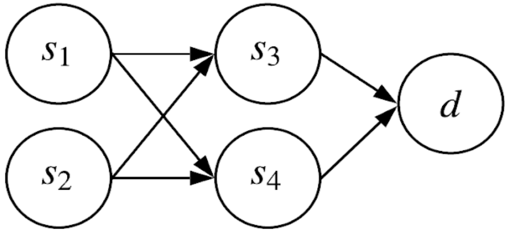
a. Captures the trade-off between detection accuracy and resource efficiency.

b. Accounts for rate of false negatives.

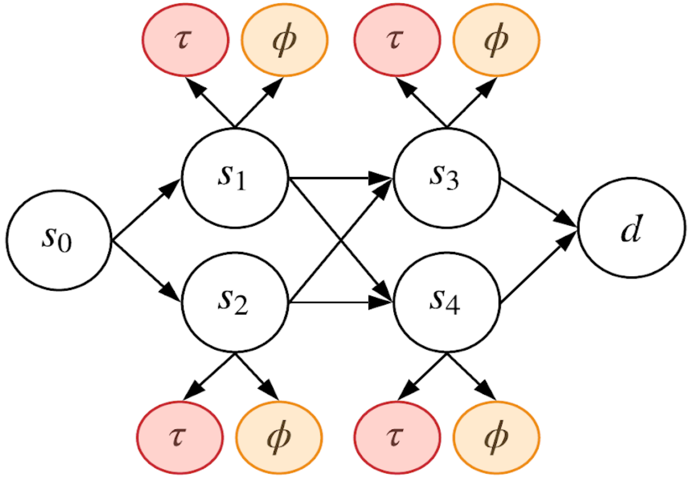
Problem Formulation

- APT **(A)** is the adversary infecting the system.
- DIFT **(D)** is the defense framework used against **A**.
- Goal of **A** is to evade detection and reach target nodes.
- Goal of **D** is to dynamically place trap locations to:
  - Maximize detection probability
  - Minimize performance overhead on system
- Strategic interactions between **D** and **A** form a game.

State space and Actions



Information Flow Graph



State Space

τ :Trap State
ϕ :Drop State

$$\mathbf{S} = \{s_0, s_1, \ldots, s_{N-1}, d, \phi, \tau\}$$

Collection of entry nodes | Node in IFGs | Terminal state | Quit state | Trapped state

We assume that both players know destination node d.

ACTIONS:

- **A** (a_t): {Transitioning to a out-neighboring state, Quit}
- **D** (d_t). **:** {Trap, no Trap}
- No actions are allowed for D at nodes in the set $\{s_0, \phi, \tau\}$

Transition Structure

DIFT (D)

$$p_D = \{p_D(s) : s \in \mathbf{S}, s \notin \{s_0, d, \phi, \tau\}\}$$

$p_D(s)$: Probability of trapping in *s*

$1 - p_D(s)$: Probability of not trapping in *s*

APT (A)

For $s \notin \{s_0, d, \tau, \phi\}$

$$p_A(s) = \{p(s, s') : s \in \mathbf{S}, s' \in N(s)\}$$

$p_A(s)$: Probability distribution of all possible actions in state *s*.

False negatives of DIFT ($FN(s_i)$) depends on:

- Number of security rules
- Strength security rules

Given actions $a_t = s_{i'}$ and $d_t = 1$. The next state is given by:

$$s_{t+1} = \begin{cases} s_{i'} & w.p. \quad FN(s_i) \\ \tau & w.p. \quad 1 - FN(s_i) \end{cases}$$

We assume policies of both players to be stationary

Payoff Structure

DIFT (D)

$$r_D(s,d,a) \begin{cases} \alpha_D > 0, if\ s = \tau & \text{Reward for detecting} \\ \beta_D < 0, if\ s = d & \text{Penalty for evading detection} \\ \sigma_D \geq 0, if\ s = \phi & \text{Reward for dropping out} \\ C_D(s) < 0, if\ s \in S\ and\ d = 1 & \text{Resource cost} \\ 0, otherwise \end{cases}$$

APT (A)

$$r_A(s,d,a) \begin{cases} \alpha_A < 0, if\ s = \tau & \text{Penalty for getting detected} \\ \beta_A > 0, if\ s = d & \text{Reward for reaching destination} \\ \sigma_A \leq 0, if\ s = \phi & \text{Penalty for dropping out} \\ 0, otherwise \end{cases}$$

Payoff Structure

$$U_A(p_D, p_A) = p_\tau^{(T)}\alpha_A + p_R^{(T)}(d)\beta_A + p_\phi^{(T)}\sigma_A$$

$$U_D(p_D, p_A) = \sum_{v_i \in V_{(G)}} (p_D(s_i)\mathcal{C}_D(v_i)) + p_\tau^{(T)}\alpha_D + p_R^{(T)}(d)\beta_D + p_\phi^{(T)}\sigma_D$$

$p_\tau^{(T)}$: Probability of being in the trapped state at terminal time T

$p_R^{(T)}$: Probability of reaching the destination at terminal time T

$p_\phi^{(T)}$: Probability of being drop state at terminal time T

Terminating Conditions

1. The adversary is trapped by the defender.
2. The adversary reaches the destination.
3. The adversary quits the game.

Payoff Structure

$$U_A(p_D,p_A) = p_\tau^{(T)}\alpha_A + p_R^{(T)}(d)\beta_A + p_\phi^{(T)}\sigma_A$$

$$U_D(p_D,p_A) = \sum_{v_i \in V_{(G)}} (p_D(s_i)\mathcal{C}_D(v_i)) + p_\tau^{(T)}\alpha_D + p_R^{(T)}(d)\beta_D + p_\phi^{(T)}\sigma_D$$

$$p_\tau(s_i) = p_D(s_i)FN(s_i)$$

$$p_R(s_i) = \sum_{s_i \in S} p_A(s_j, s_i)p_R(s_i)(1 - p_\tau(s_j))$$

$$p_\phi(s_i) = p_A(s_i, \phi)p_R(s_i)(1 - p_\tau(s_j))$$

Terminating Conditions

1. The adversary is trapped by the defender.
2. The adversary reaches the destination.
3. The adversary quits the game.

Properties of APT vs. DIFT game

Nonzero-sum	Stochastic	Imperfect Information	Incomplete Information
Defender payoff: Reward, penalty, cost of tagging Adversary payoff: Reward, penalty	Uncertainty in both players' actions	Defender cannot distinguish between benign and malicious flow	Unknown rate of false negatives

Solution Concept

Strategy pair $(p_A^*, p_D^*) \in \mathbf{P}_A \times \mathbf{P}_D$ *forms a Nash equilibrium in stochastic stationary strategies for any* $\epsilon > 0$ *and for all* $p_A \in \mathbf{p}_A$ *and* $p_D \in \mathbf{p}_D$ *if*

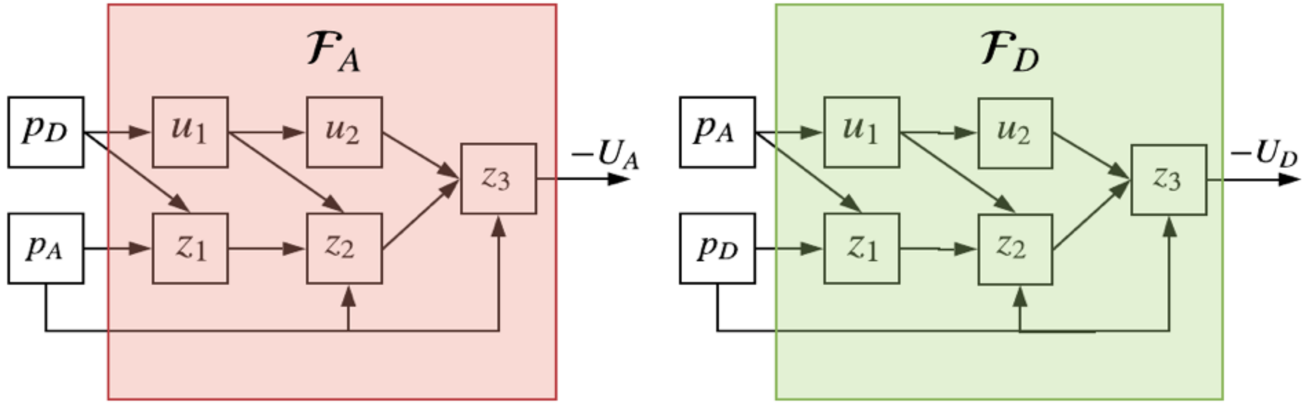
$$U_A\,(p_A^*, p_D^*) \geq (p_A, p_D^*)$$
$$U_D(p_A^*, p_D^*) \geq (p_A^*, p_D\,)$$

Result 1:

The APT vs. DIFT game satisfies the following:

1. The game terminates in at most $N + 3$ number of steps.
2. There exists a Nash Equilibrium (NE) for the game.
3. For a given strategy pair (p_A, p_D) computation of $p_R^{(T)}$, $p_\tau^{(T)}$ and $p_\phi^{(T)}$ has complexity **linear** in the number of states and edges in S.

Modified Partially Input Convex Neural Networks (PICNNs)

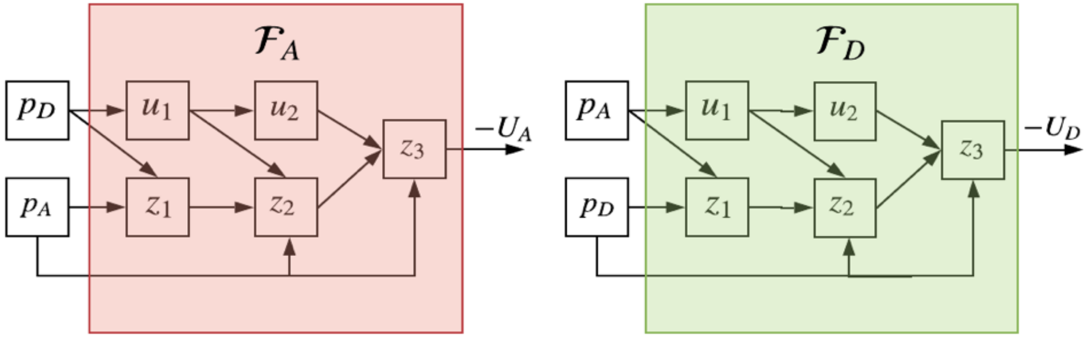


The modified PICNN model defines a neural network with k layers over output y by implementing the following architecture for $i = 0,\ldots,k$

$$z_{i+1} = g_i\left(W_i^{(z)} z_i + W_i^{(y)} y + W_i^{(u)} u_i + b_i\right), f(y;\theta) = z_k, u_0 = x$$

B. Amos, L. Xu, and J. Z. Kolter, “Input convex neural networks,” in Proceedings of the 34th International Conference on Machine Learning, vol. 70. JMLR. org, 2017, pp. 146–155.

Results



$$z_{i+1} = g_i\left(W_i^{(z)} z_i + W_i^{(y)} y + W_i^{(u)} u_i + b_i\right), f(y;\theta) = z_k, u_0 = x$$

Result 2:

The function $\mathcal{F}$ is convex in y provided that all $W_{(1:k-1)}^{(z)}$ are non-negative, and all functions g_i are convex and non-decreasing.

B. Amos, L. Xu, and J. Z. Kolter, “Input convex neural networks,” in Proceedings of the 34th International Conference on Machine Learning, vol. 70. JMLR. org, 2017, pp. 146–155.

Supervised Learning for Games

Data Generation → **Training Partial Input Convex Neural Networks (PICNN)** → **Alternating Optimization**

Generate training samples (p_A, p_D) and corresponding labels U_A, U_D

Train two partially input convex neural networks, $\mathcal{F}_A, \mathcal{F}_D$ to predict U_A, U_D respectively for a given (p_A, p_D)

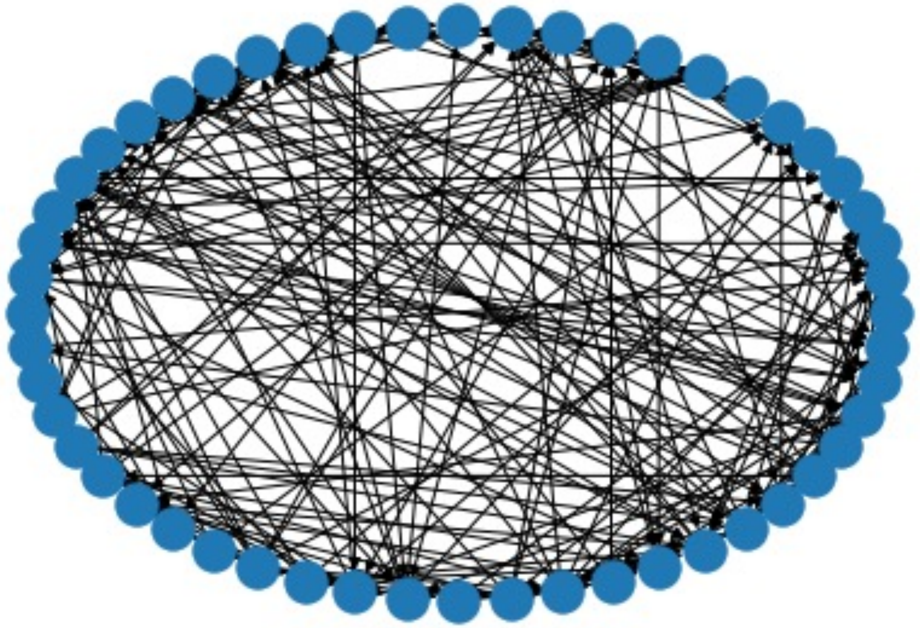
1. Update the strategy of each player by fixing the strategy of the opponent.

2. Optimize player strategy by maximizing the corresponding payoff function.

Case Studies

To demonstrate the performance of algorithm, we conducted two different simulation studies: (i) using a randomly generated IFG and (ii) using an IFG of ScreenGrab attack data obtained by the Refinable Attack Investigation System (RAIN)

Case Study: Synthetic Graph



- Number of nodes = 50
- Erdös Renyi graph with directed edge probability p = 0.2
- Destination node (d) is the 50th node.
- Resource cost is generated from a uniform distribution [0,10].

Case Studies

We conducted a sensitivity analysis to validate that the converged strategies correspond to a Nash Equilibrium of the game.

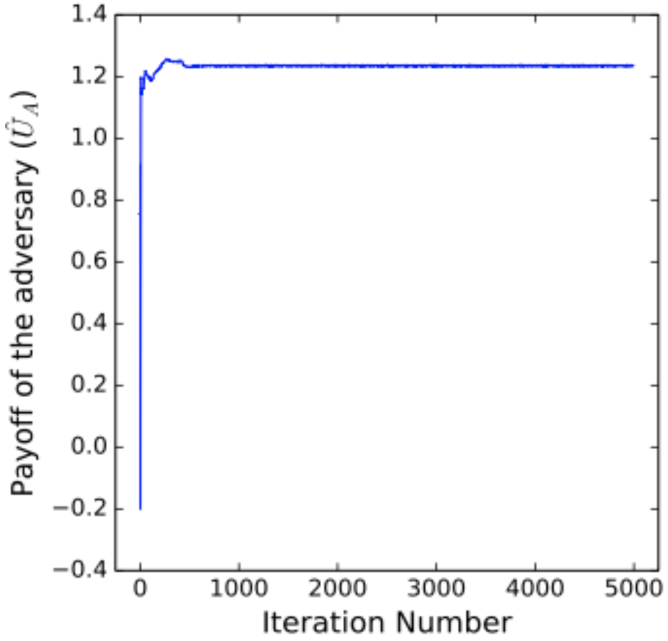
We first initialize the strategies of both players to the strategy returned by algorithm. Then we perturb the adversary's strategy while keeping the defender's strategy fixed. Similarly, we perturb the defender's strategy while keeping the adversary strategy fixed. Here, the rationale is that if the strategies returned by algorithm is a NE (with respect to the approximated payoff functions), then perturbing the strategy of a player should not improve its payoff. Therefore, the payoff from perturbed strategies should be equal to or less than that of the output of the algorithm.

The results provided below are obtained by generating 100 different perturbations of each player's strategy while keeping other player's strategy fixed.

Simulation Results

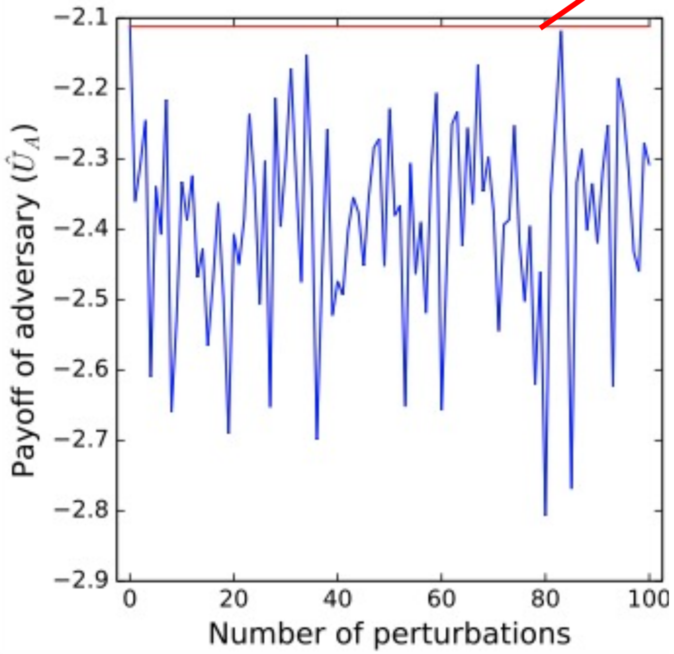
Payoff received from at the ϵ-NE obtained from the algorithm

APT's payoff vs. iteration



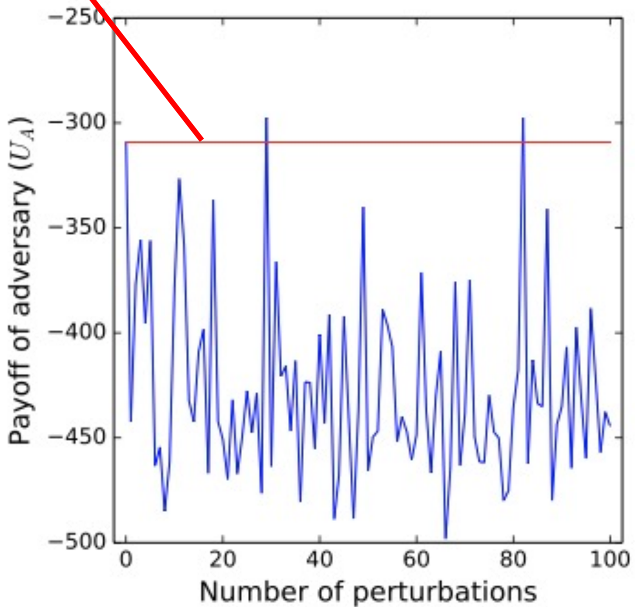
Payoff Convergence

APT's payoff for 100 perturbed strategies



Convex Approximation

APT's payoff for 100 perturbed strategies

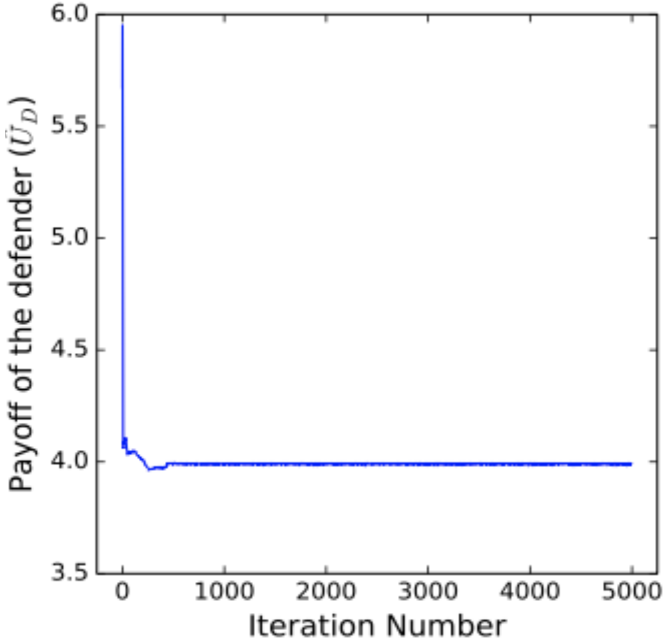


Original Function

Simulation Results

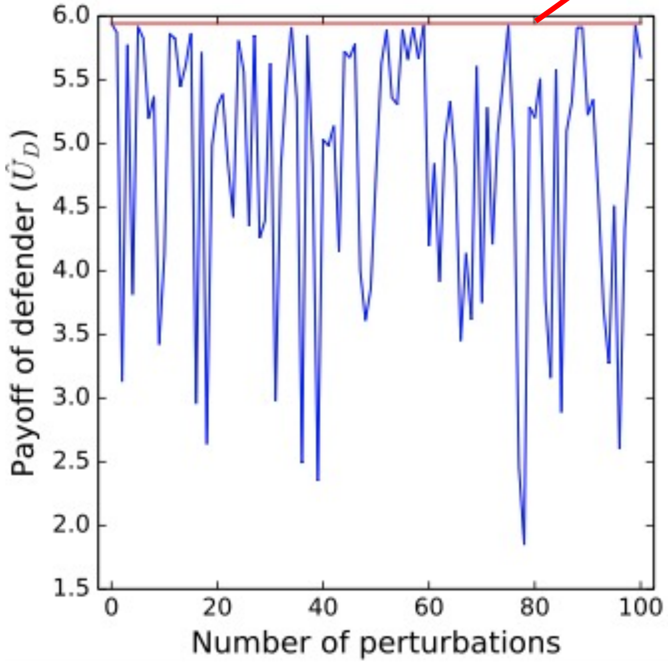
Payoff received from at the ϵ-NE obtained from the algorithm

DIFT's payoff vs. iteration



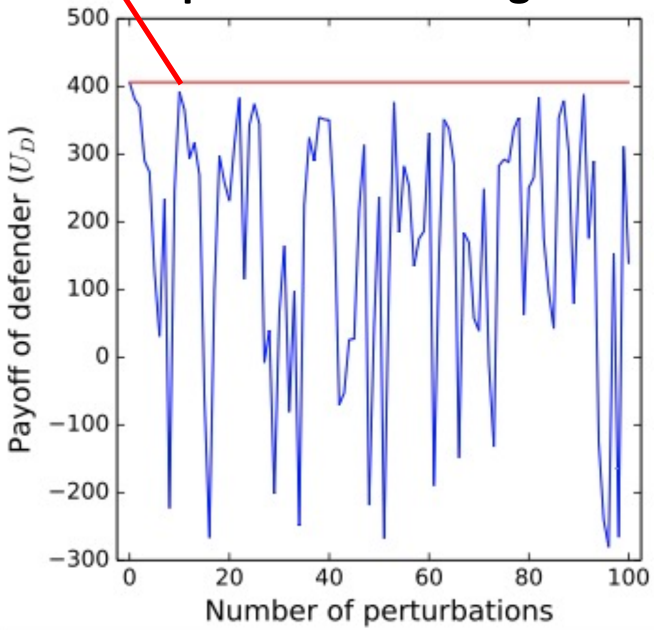
Payoff Convergence

DIFT's payoff for 100 perturbed strategies



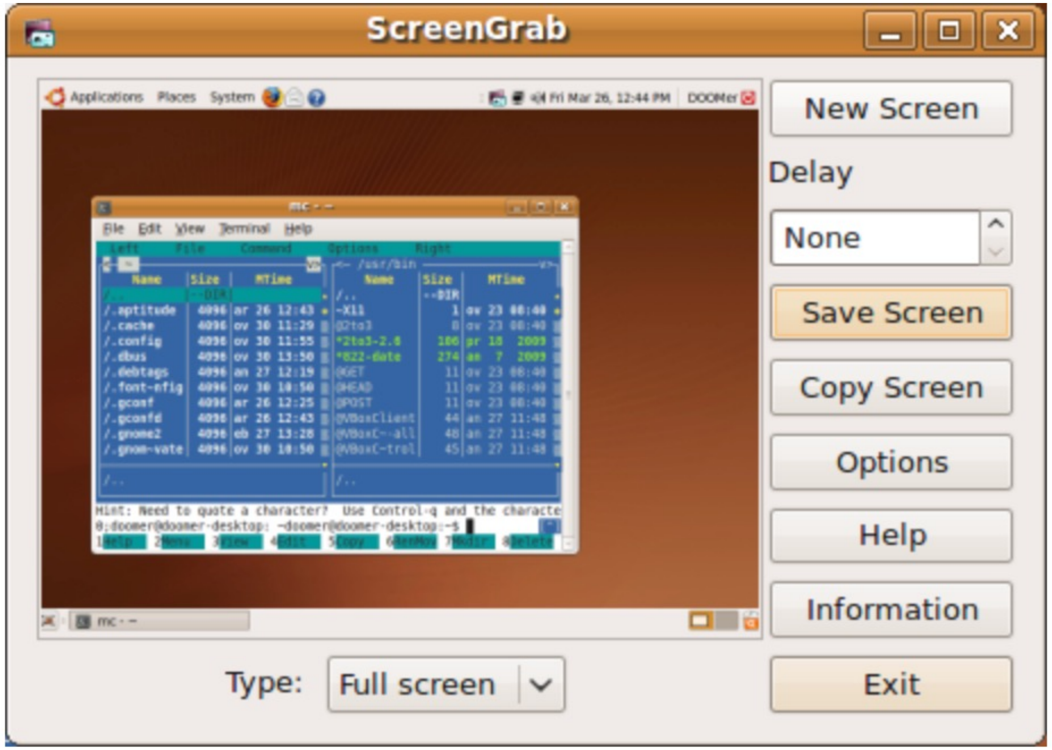
Convex Approximation

DIFT's payoff for 100 perturbed strategies



Original Function

Case Study: ScreenGrab Attack

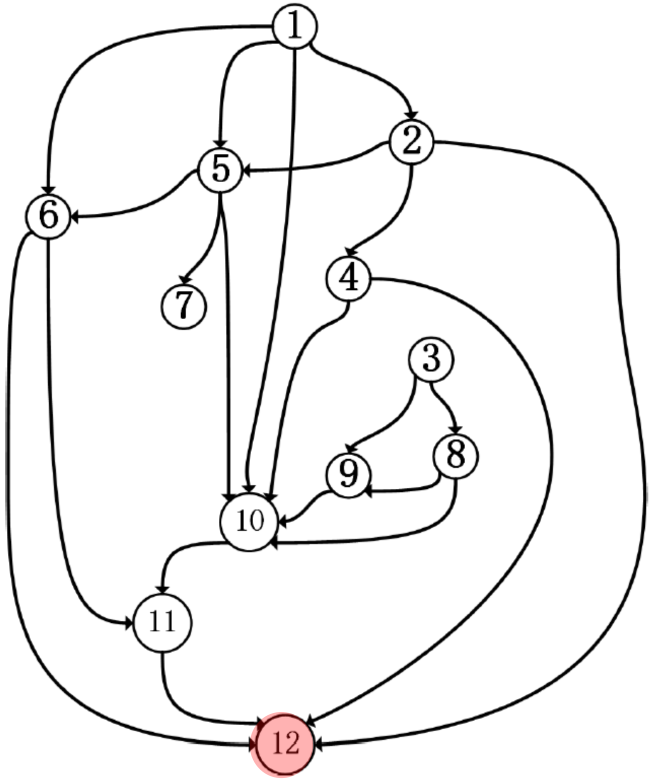


- ScreenGrab: Linux program to take screenshots.
- Adversary wants to gain access to ScreenGrab process
- Fraction of flows traversing through each process ($Prob(s)$) extracted from RAIN log data

Y. Ji, S. Lee, E. Downing, W. Wang, M. Fazzini, T. Kim, A. Orso, and W. Lee. "RAIN: Refinable Attack Investigation with On-demand Inter-Process Information Flow Tracking." In *ACM SIGSAC CCS*, 2017.

Case Study: ScreenGrab Attack

Pruned IFG for the ScreenGrab Attack



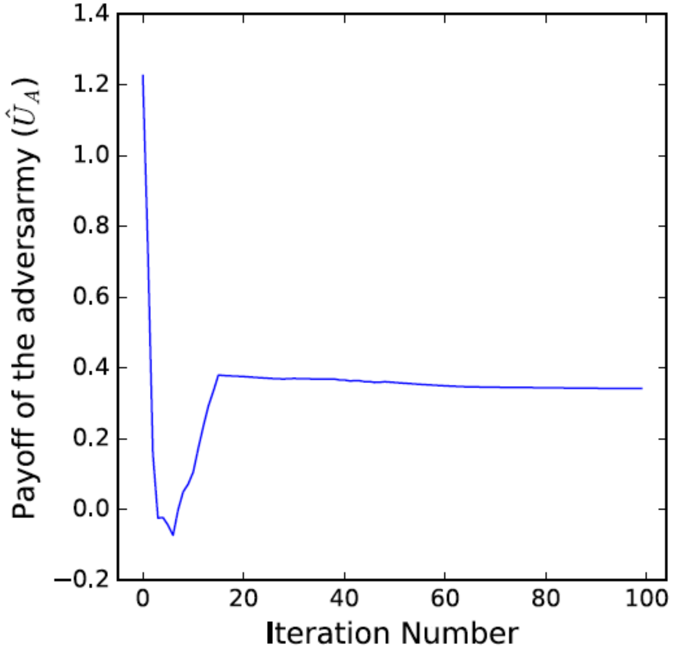
- ScreenGrab: Linux program to take screenshots.
- Adversary wants to gain access to ScreenGrab process (node 12)
- Fraction of flows traversing through each process ($Prob(s)$) extracted from RAIN log data

Y. Ji, S. Lee, E. Downing, W. Wang, M. Fazzini, T. Kim, A. Orso, and W. Lee. "RAIN: Refinable Attack Investigation with On-demand Inter-Process Information Flow Tracking." In *ACM SIGSAC CCS*, 2017.

Simulation Results

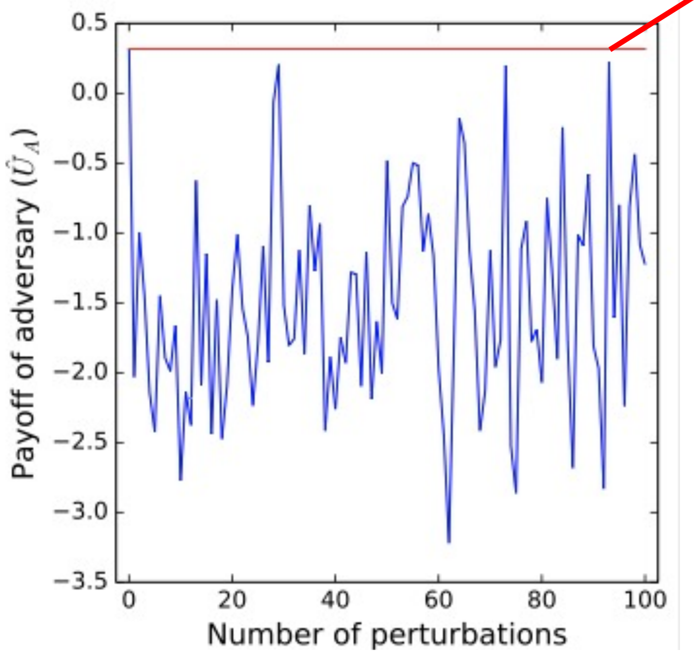
Payoff received from at the ϵ-NE obtained from the algorithm

APT's payoff vs. iteration



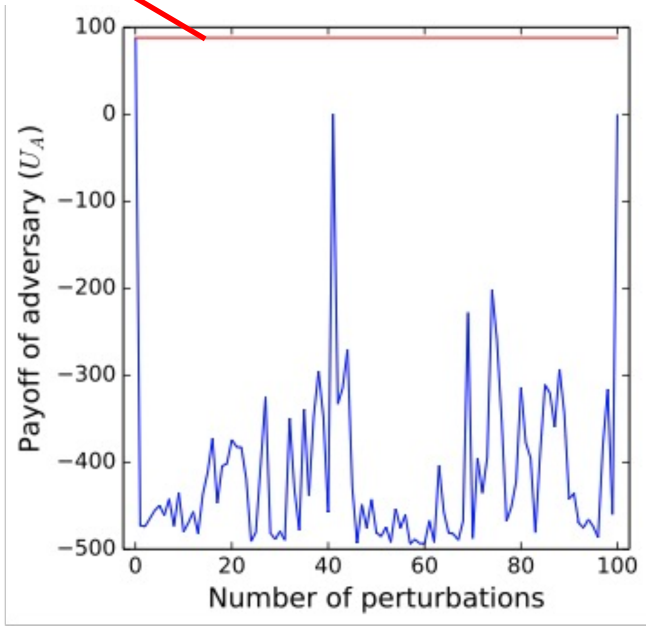
Payoff Convergence

APT's payoff for 100 perturbed strategies



Convex Approximation

APT's payoff for 100 perturbed strategies

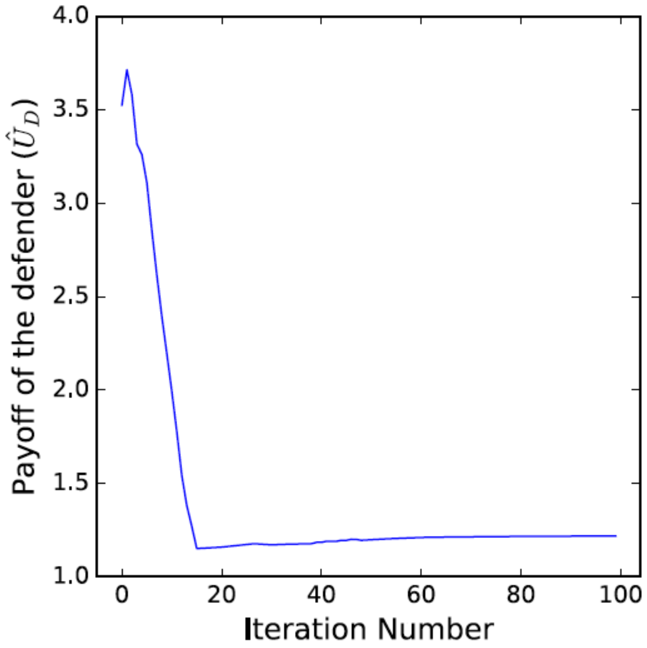


Original Function

Simulation Results

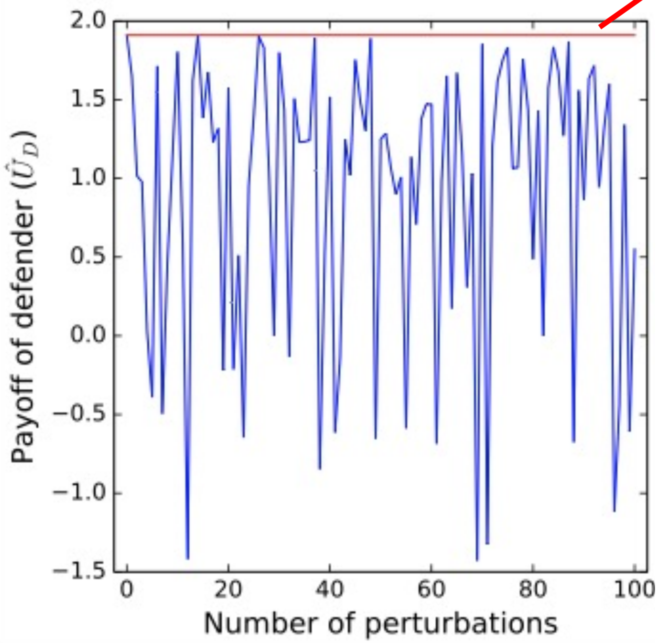
Payoff received from at the ϵ-NE obtained from the algorithm

DIFT's payoff vs. iteration



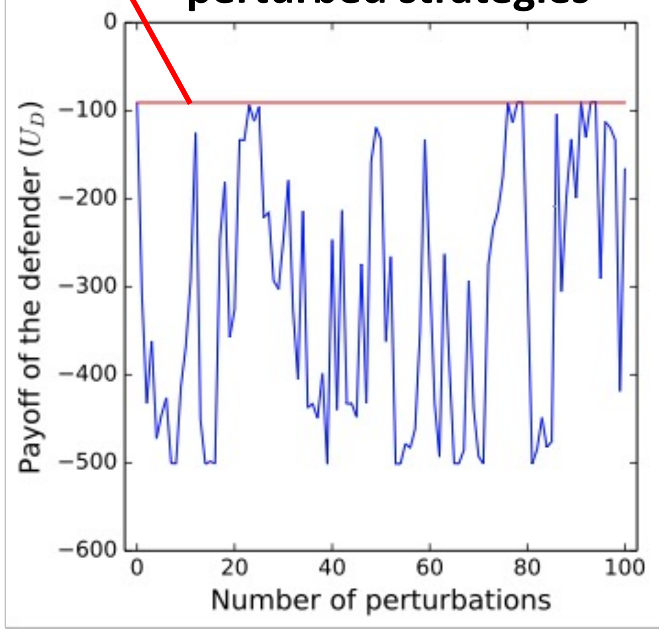
Payoff Convergence

DIFT's payoff for 100 perturbed strategies



Convex Approximation

DIFT's payoff for 100 perturbed strategies



Original Function

Observations

As can be seen, for most of the random perturbations, the payoff obtained by the non-neural network approach is indeed less than the payoff of the strategy returned by the neural network.

Specifically, in random graph experiment and for p_A, only two of the 100 perturbations resulted in higher payoff and for p_D, no perturbation yielded better payoff. Moreover, in ScreenGrab experiment, no perturbation resulted in higher payoff for either p_A or p_D. These results empirically validate that the proposed approach learns an approximate equilibrium.

Conclusion

- Formulated the interaction between APTs and DIFT as a two-player, multi-stage stochastic game with incomplete and imperfect information.

- Presented a supervised learning-based algorithm to learn an approximate Nash equilibrium for the game when the transition probabilities are unknown.

Future Work

- Characterize the convex approximation factor for the payoff functions of both players.
- Analyze the trade-off between obtaining a good convex approximation vs. the accuracy of the partial input convex neural networks.
- Investigate and apply the supervised learning approach to other types of games.

Thank You !

Email : shrm145@uw.edu

Project ADAPT website: https://adapt.ece.uw.edu/