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

Abstract— **Dynamic Information Flow Tracking (DIFT) has been proposed to detect stealthy and persistent cyber attacks in a computer system that evade existing defense mechanisms such as firewalls and signature-based antivirus systems. A DIFT-based defense tracks the propagation of suspicious information flows across the system and dynamically generates security analysis to identify possible attacks, at the cost of additional performance and memory overhead for analyzing non-adversarial information flows. In this paper, we model the interaction between adversarial information flows and DIFT on a partially known system as a nonzero-sum stochastic game. Our game model captures the probability that the adversary evades detection even when it is analyzed using the security policies (false-negatives) and the performance overhead incurred by the defender for analyzing the non-adversarial flows in the system. We prove the existence of a Nash equilibrium (NE) and propose a supervised learning-based approach to find an approximate NE. Our approach is based on a partially input convex neural network that learns a mapping between the strategies and payoffs of the players with the available system knowledge, and an alternating optimization technique that updates the players' strategies to obtain an approximate equilibrium. We evaluate the performance of the proposed approach and empirically show the convergence to an approximate NE for synthetic random generated graphs and real-world dataset collected using Refinable Attack INvestigation (RAIN) framework.**

I. Introduction

Advanced Persistent Threats (APTs) have emerged as a security threat to organizations, including national defense, manufacturing, and the financial industry [1]. APTs are distinct from the traditional cyber attacks in the following aspects: first, they use customized incursion techniques and have specific targets with the goal to gather confidential data and sabotage critical infrastructures. Second, they adopt prolonged and stealthy attacking strategies to cause more permanent, significant, and irreversible damages. Third, they are methodically designed to bypass conventional security mechanisms. Hence, detection of APTs is typically hard.

Despite the fact that APTs are stealthy, interaction of APTs with the system during the attack introduce information flows that include data-flow and control-flow commands. Since information flows are recorded in the system log, analyzing the suspicious information flows is a feasible technique to detect the presence of adversaries. Dynamic Information Flow Tracking (DIFT) [2] is a flow tracking-based mechanism that is widely used to detect APTs. 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. An optimal selection of processes, however, also depends on the interaction of the adversary with the system.

In this paper, we model a cost effective DIFT-based detection mechanism against APTs. The effectiveness of DIFT at each stage of the attack depends on the actions of the APT. Further, the probability of APT evading detection depends on the processes at which DIFT perform security analysis. We model the strategic interaction between APTs and DIFT as a a dynamic game that captures the trade-off between detection efficiency and resource efficiency. The game evolves on the Information Flow Graph (IFG) of the system constructed from the system log, where at every stage of the game the APT chooses the next operation and DIFT decides whether to conduct a security analysis on the flow or not.

The APT vs. DIFT game is nonzero-sum due to different parameter values and the additional resource cost on DIFT. Moreover, the game has an asymmetric information structure as DIFT is incapable of distinguishing a malicious flow or benign flow. The position of the information flow in the IFG along with actions chosen by APT and DIFT yield a probability distribution on the allowed state transitions in the game. The transition probabilities are determined by the rate of false negatives generated by the system at different processes and in practical scenarios are unknown. Thus both DIFT and the APT have partial knowledge about the system. We propose a supervised learning-based approach to learn an approximate Nash equilibrium (NE) of the game which implicitly learns the unknown transition probabilities.

We make the following contributions in this paper:

- We formulate a nonzero-sum stochastic game that captures the interaction between the DIFT defense and adversarial flows in a system with partial knowledge. The game captures the information asymmetry among players and false negatives generated by DIFT.
- We prove that the undiscounted game terminates in finite number of steps and the game has a NE.
- We propose a supervised learning-based approach that utilizes a Partially Input Convex Neural Network

S. Misra, S. Moothedath, H. Hosseini, L. Bushnell, and R. Poovendran are with the Department of Electrical and Computer Engineering, University of Washington, Seattle, WA 98195 USA. {shrm145, sm15,hosseinh, lb2, rp3}@uw.edu.

J. Allen and W. Lee are with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA. jallen309@gatech.edu, wenke@cc.gatech.edu.

This work was supported by ONR grant N00014-16-1-2710 P00002 and DARPA TC grant DARPA FA8650-15-C-7556.

(PICNN) architecture. We use two PICNNs to learn a mapping between the strategies and the payoff functions of the players, which implicitly learn the unknown transition probabilities, in a partially known system. Then we use an alternating optimization approach to update the players' strategies to obtain an approximate NE.
- We implement our approach on data collected using Refinable Attack INvestigation (RAIN) system and empirically show the convergence of the algorithm.

This paper is organized is as follows: Section II summarizes the related work. Section III elaborates on information flow graph, and the attacker and defender models. Section IV presents the formulation of the nonzero-sum stochastic game. Section V presents some preliminary results and the proposed supervised learning-based approach. Section VI illustrates the results and discussions on the experiments conducted. Section VII concludes the paper.

II. Related Work

Stochastic games model the interaction in a dynamical system and are well studied in the context of security games [4], [5], economic games [6], and resilience of cyber-physical systems [7]. Stochastic games are formulated in [8], [9] to model the interaction between malicious attackers and the Intrusion Detection System (IDS). The network security configuration problem in distributed IDS is modeled as a nonzero-sum stochastic game and a value iteration-based algorithm to find an ε-NE is proposed in [10]. In contrast, this paper focuses on detection of APTs in a system.

A dynamic game model is given in [11] to detect APTs that can adopt adversarial deceptions. A two-level evasion-detection game model is given in [12] for detection of APTs. In our framework, we model the detection of APTs using DIFT as a dynamic stochastic game. Some effort has been made to model the detection problem of APTs using DIFT as a game [13], [14]. The game models in [13], [14] are non-stochastic as the notion of false negatives is not considered. Recently, a stochastic model of DIFT-games is proposed in [15] where the notion of conditional branching in programs is addressed. An assumption of the approach given in [15] is that the transition probabilities are assumed to be known. In this paper, we propose a novel method to collect samples and learn the unknown transition probabilities. An adaptive policy approach using a regularized Lagrange function is proposed in [16] for zero-sum games with unknown transition probabilities and irreducible state space. In contrast, the game considered in this paper is nonzero-sum and not irreducible.

Multi-agent reinforcement learning (MARL) algorithms are proposed in the literature to obtain NE strategies of nonzero-sum stochastic games when the game information such as transition and payoff functions of the players are unknown. However, algorithms with guaranteed convergence are available only for special cases. A Nash-Q learning algorithm is given in [17] that converges to a NE of a general-sum games when the NE is unique. Later [18] defined two properties, *rationality* and *convergence* that are necessary for an algorithm to converge to a NE and proposed a WOLF-policy hill climbing algorithm that is empirically shown to have good performance. An MARL algorithm is proposed and empirically shown to obtain a correlated equilibrium of a nonzero-sum game in [19]. Notice that while references [17]-[19] address nonzero-sum games with discounted reward, this paper aims to learn NE in average reward. Moreover, in our game NE is non-unique and hence the existing MARL approaches do not guarantee convergence.

III. Preliminaries

A. Information Flow Graph

Information flow graphs (IFG) are widely used by analysts for an effective cyber response [3], [20]. IFGs provide a graphical representation of the system work-flow, where the nodes in the graph form the processes, files, network connections, and memory objects in the system. Edges correspond to the system calls and are oriented in the direction of the information flows and/or causality [20]. Let directed graph $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}})$ represent IFG of the system, where $V_{\mathcal{G}} = \{v_1, \ldots, v_N\}$ and $E_{\mathcal{G}} \subseteq V_{\mathcal{G}} \times V_{\mathcal{G}}$. Given a system log, one can build the corresponding IFG which represents the lineage of the system operation. We consider directed acyclic IFGs rendered using the node versioning technique given in [20]. Our work relies on the IFG of the system to identify the entry points of the attack and its system-wide impact.

B. Attacker Model

We consider multi-step cyber-attacks called APTs. The perpetrators of these attacks remain in the system for long periods, while exploring the organization's IT infrastructure and exfiltrating or compromising critical data [20]. Let $\mathcal{P}_A$ denote an APT trying to attack a system. APTs are intelligent adversaries with specific targets, say a sensitive file in the system. We denote the target of the APT as d, where $d \in V_{\mathcal{G}}$, and refer to it as the *destination*. Further, let $\lambda \subseteq V_{\mathcal{G}}$ be the vulnerable locations in the system that susceptible to be exploited by the adversary. Adversary can enter into the system through some node in set λ and then executes operations, i.e., transitions in $\mathcal{G}$, so as to reach d before getting detected by the system.

C. Defender Model

To secure the system against APTs we employ a DIFT based detection mechanism [2]. DIFT tracks the system calls to detect the malicious information flows from an adversary and to restrict the use of these malicious flows. DIFT consists of three main components: (i) tag sources, (ii) tag propagation rules, and (iii) tag sinks (traps). Tag sources are the suspicious locations in the system, such as keyboards, network interface, and hard disks, that are tagged/tainted as spurious by DIFT. Tags are single bit or multiple bit markings depending on the level of granularity manageable with the available memory and resources. All the processed values of the tag sources are tagged and DIFT tracks the propagation of the tagged flows. When anomalous behavior is detected, tagged flows are inspected by DIFT at specific locations, referred to as *traps*. At these traps, DIFT conducts fine grain analysis to detect the attack and to perform risk assessment.

While tagging and trapping using DIFT is a promising detection mechanism against APTs, DIFT introduces memory and performance overhead on the system. Performing security analysis (trapping) of tagged flows uses considerable amount of memory of the system [21]. Thus there is a tradeoff between log granularity and system performance [3].

IV. Problem Formulation

We formulate a two-player game between the defender player, $\mathcal{P}_D$, and the adversarial player, $\mathcal{P}_A$. The state of the game at a time step t represents the position of the tagged information flow that is analyzed by DIFT. Let $\mathbf{S} = \{s_0, s_1, \ldots, s_N, \phi, \tau\}$ be the state space of the game, where $s_1, \ldots, s_N$ corresponds to the nodes $v_1, \ldots, v_N$, respectively, of the information flow graph $\mathcal{G}$. Without loss of generality, let v_N corresponds to the destination (target) node of the adversary, i.e., $s_N = d$ and we can denote the state space as $\mathbf{S} = \{s_0, s_1, \ldots, s_{N-1}, d, \phi, \tau\}$. We assume that both players know the destination node, d. In this paper, we use s_N and d interchangeably. Here, ϕ corresponds to the state when the tagged flow drops out by abandoning the attack and τ corresponds to the state when the adversary is detected. The state s_0 corresponds to a virtual state introduced into the game in order to denote the starting point of the game. In the state space $\mathbf{S}$, s_0 is connected to all nodes in λ. Thus s_0 is the state of the game at $t = 0$.

The action space of the players are discrete and finite. While $\mathcal{P}_A$ chooses actions at states in $\mathbf{S}$ so as to transition through $\mathcal{G}$ and reach d, $\mathcal{P}_D$ makes a decision to trap or not to trap the tagged flow at a state. Let $\bar{s}_t$ denote the state of the game at time instant t. Further, let $\mathcal{A}_A(\bar{s}_t)$ and $\mathcal{A}_D(\bar{s}_t)$ denote the action sets of $\mathcal{P}_A$ and $\mathcal{P}_D$, respectively, at $\bar{s}_t$. Then, for $\bar{s}_t \notin \{d, \phi, \tau\}$, $\mathcal{A}_A(\bar{s}_t) = \{v_j : (v_i, v_j) \in E_{\mathcal{G}}, \bar{s}_t = v_i\} \cup \{\varnothing\}$, where $\varnothing$ is the action of dropping out. At $\bar{s}_t = s_0$, $\mathcal{A}_A(\bar{s}_t) = \lambda$. The defender's action set at $\bar{s}_t \notin \{s_0, d, \phi, \tau\}$ is given by $\mathcal{A}_D(\bar{s}_t) = \{0, 1\}$, where 0 denotes trapping the flow and 1 otherwise. At $\bar{s}_t \in \{d, \phi, \tau\}, \mathcal{A}_A(\bar{s}_t) = \mathcal{A}_D(\bar{s}_t) = \emptyset$. The set $\{d, \phi, \tau\}$ denotes the terminal states of the game. At every non-terminal state in the game the adversary chooses a neighboring node in the IFG to transition to or to drop the attack. On the other hand, the defender chooses whether to trap the flow or not.

The defender observes the incoming tagged information flow, however, cannot distinguish if it is a benign flow or a malicious flow. As a result, even though defender knows the state of the game, it has only partial information about the state. On the other hand, the adversary does not know if the tagged flow will get trapped by the defender while choosing a transition. Thus the game is an imperfect information game.

Consider a tagged flow incoming at node $v_i \in V_{\mathcal{G}}$ at time t. Then $\bar{s}_t = s_i$. Let the action chosen by the defender and the adversary at $\bar{s}_t$ be d_t and a_t, respectively. If $d_t = 0$, then $\bar{s}_{t+1} = a_t$. That is, if defender chooses not to trap an information flow, then the flow proceeds to the node in $\mathcal{G}$ chosen by the adversary. If the defender chooses to trap the flow, then the flow is terminated with probability $\mathbb{P}(\bar{s}_t)$ and the flow transition to the state corresponding to the action of the adversary with the remaining probability. That is, if $d_t = 1$, then $\bar{s}_{t+1} = \tau$ with probability $\mathbb{P}(\bar{s}_t)$ and $\bar{s}_{t+1} = a_t$ with probability $1 - \mathbb{P}(\bar{s}_t)$. The transition probability $\mathbb{P}(\cdot)$ depends on the rate of generation of false negatives at the different nodes in the IFG. Note that, different nodes in IFG have different capabilities to perform security analysis and depending on that the value of $\mathbb{P}(\cdot)$ is different. The numerical values of these transition probabilities depend on the type of the attack and hence are in a practical setting is unknown to the analysts. As APTs are tailored attacks that can manipulate the system operation and evade conventional security mechanisms such as firewalls, anti-virus softwares, and intrusion-detection systems, $\mathbb{P}(\cdot)$'s are unknown.

At every time step, both $\mathcal{P}_A$ and $\mathcal{P}_D$ strategize for their own benefit. We denote the policy space of the attacker by $\mathbf{P}_A$ and that of the defender by $\mathbf{P}_D$. Then, $\mathbf{P}_A : \mathbf{S} \to [0,1]^{|\mathcal{A}_A|}$ and $\mathbf{P}_D : \mathbf{S} \to [0,1]^{|\mathcal{A}_D|}$. Let $p_A \in \mathbf{P}_A$ and $p_D \in \mathbf{P}_D$. For $p_D = [p_D(s_1), \ldots, p_D(s_{N-1})]$, the flow is trapped at state $\bar{s}_t$ with probability $p_D(\bar{s}_t)$ and not trapped with probability $1 - p_D(\bar{s}_t)$. For $p_A = [p_A(s_0), p_A(s_1), \ldots, p_A(s_{N-1})]$, $p_A(\bar{s}_t)$ is a probability distribution over all possible actions of the adversary at $\bar{s}_t$.

The payoff functions of the players consist of three main components. The defender earns a reward $\alpha_D > 0$ if it detects the adversary, incurs a penalty $\beta_D < 0$ if it fails to detect the adversary, and earns a reward $\sigma_D \geqslant 0$ if the adversary abandons the attack by dropping out. The adversary incurs a penalty $\alpha_A < 0$ if it gets detected by the defender, earns a reward $\beta_A > 0$ if it reaches the destination, and incurs a penalty $\sigma_A \leqslant 0$ for abandoning the attack by dropping out. Additionally, the defender also incurs a resource cost $\mathcal{C}_D(v_i)$ for conducting security analysis on benign flows at node $v_i \in V_{\mathcal{G}}$. This cost depends on the computational power and busyness of the node, as the defender needs to avoid over loading the busy processes to avoid performance and memory overhead. Notice that the game is nonzero-sum.

Let the payoff of player k at a terminal state state $\bar{s}_t$ be $c^k(\bar{s}_t)$ and at a non-terminal state $\bar{s}_t$ with action pair (a_t, d_t) be $r^k(\bar{s}_t, a_t, d_t)$, where $k \in \{A, D\}$. Then,

$$c^k(\bar{s}_t) = \begin{cases} \alpha_k, & \bar{s}_t = \tau \\ \beta_k, & \bar{s}_t = d \\ \sigma_k, & \bar{s}_t = \phi \end{cases}$$

$$r^k(\bar{s}_t, a_t, d_t) = \begin{cases} 0, & \bar{s}_t \notin \{d, \phi, \tau\} \text{and } d_t = 0 \\ 0, & \bar{s}_t \notin \{d, \phi, \tau\}, k = A, \text{and } d_t = 1 \\ \mathcal{C}_D(\bar{s}_t), & \bar{s}_t \notin \{d, \phi, \tau\}, k = D, \text{and } d_t = 1. \end{cases}$$

At each stage in game, $\bar{s}_t$ at time t, both players simultaneously choose their action a_t and d_t and receive payoffs $r^A(\bar{s}_t, a_t, d_t)$ and $r^D(\bar{s}_t, a_t, d_t)$, respectively, and transition to a next state $\bar{s}_{t+1}$. This is continued until they reach a terminal state and incur $c^A(\bar{s}_t)$ and $c^D(\bar{s}_t)$, respectively.

Now we prove a preliminary result of the game.

Theorem IV.1. *The stochastic game between the adversary and the defender terminates in at most $N + 3$ number of steps.*

Proof. Consider any arbitrary strategy pair (p_A, p_D). We prove the result using the acyclic property of the state space of the game under (p_A, p_D). The state space $\mathbf{S}$ is constructed by augmenting the IFG with states s_0, ϕ and τ. Note that a state $s \in \{d, \phi, \tau\}$ does not lie in a cycle in $\mathbf{S}$ as s is a terminal state and hence have no outgoing edge. The state s_0 does not lie in a cycle in $\mathbf{S}$ as there are no incoming edges to s_0. This concludes that a state $s \in \{s_0, d, \phi, \tau\}$ is not part of a cycle in $\mathbf{S}$. Thus a cycle can possibly exist in $\mathbf{S}$ only if there is a cycle which has states in $s_1, \ldots, s_{N-1}$. Recall that states $s_1, \ldots, s_{N-1}$ correspond to nodes $v_1, \ldots, v_{N-1}$ of $\mathcal{G}$. As $\mathcal{G}$ is acyclic, there are no cycles in $\mathbf{S}$. Since $\mathbf{S}$ is acyclic under any arbitrary strategy pair (p_A, p_D) and the state space has finite cardinality, the game terminates in finite number of steps. Further, since $|\mathbf{S}| = N+3$, $T \leqslant N+3$. □

Let T denote termination time of the game. By Theorem IV.1, $T \leqslant N+3$. Thus our game is a finite horizon game. Let U_A and U_D denote the payoff functions of $\mathcal{P}_A$ and $\mathcal{P}_D$, respectively. As the initial state of the game is s_0, for a strategy pair (p_A, p_D) the expected payoffs of the players are

$$U_A(p_A, p_D) = \mathbb{E}_{s_0, p_A, p_D}\left[\sum_{t=0}^{T}(R_t^A)\right] \text{and} \tag{1}$$

$$U_D(p_A, p_D) = \mathbb{E}_{s_0, p_A, p_D}\left[\sum_{t=0}^{T}(R_t^D)\right], \tag{2}$$

where $\mathbb{E}_{s_0, p_A, p_D}$ denotes the expectation with respect to s_0, p_A, and p_D and

$$R_t^k = \begin{cases} r^k(\bar{s}_t, a_t, d_t), & t < T \\ c^k(\bar{s}_t), & t = T. \end{cases} \tag{3}$$

Hence, the APT vs. DIFT game is a two-player multi-stage finite horizon stochastic game.

The solution concept of the game is defined as follows. We first describe the concept of a player's best response to a given mixed strategy of the opponent player.

Definition IV.2. *Let $p_D : \mathbf{S} \to [0,1]^{|\mathcal{A}_D|}$ denote a defender strategy and $p_A : \mathbf{S} \to [0,1]^{|\mathcal{A}_A|}$ denote an adversary strategy. The set of best responses of the defender is given by $BR(p_A) = \arg\max_{p_D}\{U_D(p_D, p_A) : p_D \in [0,1]^{\mathcal{A}_D}\}$. Similarly, the best responses of the adversary given by $BR(p_D) = \arg\max_{p_A}\{U_A(p_D, p_A) : p_A \in [0,1]^{\mathcal{A}_A}\}$.*

Next we define the notion of Nash equilibrium of a game. A mixed strategy profile $(p_D^\star, p_A^\star)$, where $p_D^\star \in \mathbf{P}_D$ and $p_A^\star \in \mathbf{P}_A$, is a *Nash equilibrium* (NE) if the following definition hold.

Definition IV.3. *A pair of mixed strategies $(p_D^\star, p_A^\star)$ is a* Nash equilibrium *if $p_D^\star \in BR(p_A^\star)$ and $p_A^\star \in BR(p_D^\star)$.*

Definition IV.3 translates to players selecting policies $(p_D^\star, p_A^\star) \in \mathbf{P}_D \times \mathbf{P}_A$ such that

$$U_D(p_D^\star, p_A^\star) \geqslant U_D(p_D, p_A^\star), \text{ for all } p_D \in \mathbf{P}_D \text{ and}, \tag{4}$$

$$U_A(p_D^\star, p_A^\star) \geqslant U_A(p_D^\star, p_A), \text{ for all } p_A \in \mathbf{P}_A. \tag{5}$$

Now we define the notion of ε-Nash equilibrium here.

Definition IV.4. *$(p_A^\varepsilon, p_D^\varepsilon) \in \mathbf{P}_A \times \mathbf{P}_D$ forms an ε-equilibrium in stochastic stationary strategies for any $\varepsilon > 0$ and for all $p_A \in \mathbf{p}_A$ and $p_D \in \mathbf{p}_D$ if*

$$U_A(p_A^\varepsilon, p_D^\varepsilon) \geqslant U_A(p_A, p_D^\varepsilon) - \varepsilon, \text{ and}$$
$$U_D(p_A^\varepsilon, p_D^\varepsilon) \geqslant U_D(p_A^\varepsilon, p_D) - \varepsilon.$$

V. Solution to the APT-DIFT Game

In this section, we first prove the existence of a NE of the game and few preliminary results. Then we present our learning-based approach to solve the stochastic game with unknown transition probabilities.

A. Preliminary Results

Proposition V.1. *There exists a NE for the nonzero-sum undiscounted stochastic game between APT and DIFT.*

Proof. It is shown in [22] that there exists a Nash equilibrium for a nonzero-sum stochastic game with asymmetric information structure, under stochastic behavioral policies, when the time horizon is finite. Theorem IV.1 prove that the APT vs. DIFT game will terminate in finite number of steps. Further, the behavioral strategy space is a subset of the strategy space $\mathbf{P}_A \times \mathbf{P}_D$. Hence by the result in [22], the proof follows. □

Although Proposition V.1 ensures existence of NE for our finite-horizon stochastic game, computation of a NE is challenging. Moreover, while the interaction between APTs and DIFT is modeled as a stochastic game, solving the game is challenging due to the unknown transition probabilities. To this end, we propose a learning-based approach to solve the game. We first introduce the following concepts and notations to obtain an alternate representation of the payoffs of the players. Let $p_\tau^{(t)}$, $p_R^{(t)}(s_i)$, and $p_\phi^{(t)}$ be the probabilities with which the adversarial flow is detected by the defender, the adversarial flow reaches state $s_i \in \mathbf{S}$, and the adversary drops out of the game, respectively, for a given strategy pair (p_A, p_D) at time t in the game.

Lemma V.2. *Consider the nonzero-sum, undiscounted stochastic between the APT and DIFT. Let $p_\tau^{(t)}$, $p_R^{(t)}(s_i)$, and $p_\phi^{(t)}$ be the cumulative probabilities with which the adversarial flow is detected by the defender, the adversarial flow reaches state $s_i \in \mathbf{S}$, and the adversary drops out of the game, respectively, for a given strategy pair (p_A, p_D) at time step t of the game. Then, $p_\tau^t + \sum_{s_i \in \mathbf{S}} p_R^{(t)}(s_i) + p_\phi^{(t)} = 1$.*

Proof. We prove the result using an induction argument. The induction hypothesis is that at any time step t, $p_\tau^{(t)} + \sum_{s_i \in \mathbf{S}} p_R^{(t)}(s_i) + p_\phi^{(t)} = 1$.
Base step: we consider $t = 0$ as the base step. For $t = 0$, the game state is s_0. At $t = 0$, $p_\tau^{(0)} = 0$, $p_R^{(t)}(s_0) = 1$, $p_R^{(t)}(s_i) = 0$, for all $s_i \in \mathbf{S}, s_i \neq s_0$, and $p_\phi^{(0)} = 0$. Thus $p_\tau^{(0)} + \sum_{s_i \in \mathbf{S}} p_R^{(0)}(s_i) + p_\phi^{(0)} = 1$ and this proves the base step.
Induction step: for the induction step we assume that $p_\tau^{(k)} + \sum_{s_i \in \mathbf{S}} p_R^{(k)}(s_i) + p_\phi^{(k)} = 1$.

Now we will prove that $p_\tau^{(k+1)} + \sum_{s_i \in \mathbf{S}} p_R^{(k+1)}(s_i) + p_\phi^{(k+1)} = 1$. If $k = T$, then the proof is straightforward. Hence we assume $k < T$.

$$p_\tau^{(k+1)} + \sum_{s_i \in \mathbf{S}} p_R^{(k+1)}(s_i) + p_\phi^{(k+1)} = p_\tau^{(k)} + \sum_{s_i \in \mathbf{S}} p_R^{(k)}(s_i)\Big\{$$

$$\sum_{a\in\mathcal{A}_A(s_i)} \underbrace{\Big(p_A(s_i,a)(1-p_D(s_i))+p_A(s_i,a)p_D(s_i)(1-\mathbb{P}(s_i))+}_{\text{transition to } s_j\in\mathbf{S}\setminus\{\tau\}}$$

$$\underbrace{p_A(s_i,a)p_D(s_i)\mathbb{P}(s_i)\Big)}_{\text{transition to } \tau}\Big\}+p_\phi^{(k)}$$

Note that $p_A(s_i,a)(1-p_D(s_i))+p_A(s_i,a)p_D(s_i)(1-\mathbb{P}(s_i))+p_A(s_i,a)p_D(s_i)\mathbb{P}(s_i)\Big)=p_A(s_i,a)$. Further, $\sum_{a\in\mathcal{A}_A(s_i)}p_A(s_i,a)=1$. This gives

$$p_\tau^{(k+1)}+\sum_{s_i\in\mathbf{S}}p_R^{(k+1)}(s_i)+p_\phi^{(k+1)}=p_\tau^{(k)}+\sum_{s_i\in\mathbf{S}}p_R^{(k)}(s_i)+p_\phi^{(k)}$$

By the induction step, $p_\tau^{(k+1)}+\sum_{s_i\in\mathbf{S}}p_R^{(k+1)}(s_i)+p_\phi^{(k+1)}=1$. $\square$

At the time of termination, i.e., $t=T$, the game satisfies one of the following: (i) $\bar{s}_T=\tau$, (ii) $\bar{s}_T=d$, and (ii) $\bar{s}_T=\phi$. Thus the players' total payoffs for a given strategy pair (p_A,p_D) can be alternatively represented as

$$U_A(p_D,p_A)=p_\tau^{(T)}\,\alpha^A+p_R^{(T)}(d)\,\beta^A+p_\phi^{(T)}\,\sigma_A, \tag{6}$$

$$U_D(p_D,p_A)=\sum_{v_i\in V_\mathcal{G}}\Big(p_D(s_i)\mathcal{C}_D(v_i)\Big)+p_\tau^{(T)}\alpha^D+p_R^{(T)}(d)\beta^D+p_\phi^{(T)}\sigma_D \tag{7}$$

As Eqs. (1), (2) are equivalent to Eqs. (6), (7), respectively, in our analysis we use Eqs. (6), (7). At termination, by Lemma V.2 $p_\tau^{(T)}+p_R^{(T)}+p_\phi^{(T)}=1$. We use Lemma V.2 in the implementation of our supervised learning-based approach which is described below.

B. Supervised Learning for Game

In this section, we present our approach to solve the stochastic APT vs. DIFT game when the transition probabilities are unknown. While game theory provides a framework to formulate the strategic interaction, solving the game is challenging due to the unknown transition probabilities, $\mathbb{P}$. To this end, we propose a supervised learning-based approach to solve the problem. Our approach consists of two key steps: (i) training two neural networks, $\mathcal{F}_A$ and $\mathcal{F}_D$, to predict U_A, U_D, respectively, for a given (p_A,p_D); and (ii) optimizing alternatively to find equilibrium strategies for the players.

Data Generation: In order to train the neural networks, we generate random samples of strategy pairs (p_A,p_D). We first compute the respective $p_R^{(T)}$, $p_\tau^{(T)}$, $p_\phi^{(T)}$ values using $\mathbb{P}$ and then evaluate U_A, U_D. Note that, computation of $p_R^{(T)}$, $p_\tau^{(T)}$, and $p_\phi^{(T)}$ for a given (p_A,p_D) involves all possible paths in the state space which is exponential in number. Hence we propose an alternate technique which has complexity linear in the number of states and edges in $\mathbf{S}$. To elaborate on this, we introduce the following concepts and notations.

Definition V.3 ([23]). *A topological ordering of a directed graph is a linear ordering of its vertices such that for every directed edge u,v from vertex u to vertex v, u comes before v in the ordering.*

For a directed graph with vertex set V and edge set E there exists an algorithm of complexity $O(|V|+|E|)$ to find the topological order [23]. Let $\mathcal{S}$ be the topologically ordered set of nodes of $\mathbf{S}\setminus\{\phi,\tau\}$. Without loss of generality, we assume $\mathcal{S}=\{s_0,s_1,\ldots,s_{N-1},d\}$ is in topological order.

For a given strategy (p_A,p_D), let $p_\tau^{(T)}(s_i)$ be the probability that a tagged flow at state s_i gets detected.

$$p_\tau^{(T)}(s_i)=p_D(s_i)\mathbb{P}(s_i) \tag{8}$$

Then, the probability that the adversarial flow reaches state s_i is given by,

$$p_R^{(T)}(s_i)=\sum_{s_j\in\mathcal{S}}p_A(s_j,s_i)p_R^{(T)}(s_j)(1-p_\tau^{(T)}(s_j)). \tag{9}$$

Lemma V.4. *Let $\mathbf{S}$ be the state space of the APT vs. DIFT game. For a given strategy pair (p_A,p_D), computation of $p_R^{(T)}(d),p_\tau^{(T)},p_\phi^{(T)}$ has complexity linear in the number of states and edges in $\mathbf{S}$.*

Proof. We begin the computation of $p_R^{(T)}(s_i)$'s in the order specified in $\mathcal{S}$. Note that $p_R^{(T)}(s_i)$ depends on s_j's where s_j appear before s_i in the topologically ordered list $\mathcal{S}$. Therefore, the values of $p_R^{(T)}(s_j)$'s in Eq. (9) are already computed before $p_R^{(T)}(s_i)$ is evaluated. Given topological order $\mathcal{S}$, Eqs. (8) and (9) can be computed recursively in $|\mathcal{S}|$ number of operations. The number of operations required to find the list $\mathcal{S}$ is linear in $|\mathbf{S}|$ and the number of edges in $\mathbf{S}$ [23]. Thus the proof follows. $\square$

Given samples of (p_A,p_D), we can compute values of U_A, U_D using Eqs. (8) and (9) and Lemma V.2.

Training and Alternating Optimization: To compute a NE of the game with unknown transition probabilities, we first train two neural networks $\mathcal{F}_A$ and $\mathcal{F}_D$ with the generated data set. Since there is no closed form solution for the joint optimization of the nonzero-sum game, we update the player strategies via an alternating optimization approach to obtain an approximate NE. The key steps of this approach are:

(1) train $\mathcal{F}_A,\mathcal{F}_D$ to predict U_A,U_D, respectively, for an input (p_A,p_D) and
(2) alternately update p_A (p_D, resp.) such that U_A (U_D, resp.) is maximized keeping p_D (p_A, resp.) fixed.

The alternating optimization converges when neither U_A nor U_D change any further. However, note that U_A and U_D are non-concave functions in p_A, p_D, respectively. Hence the resultant policies (p_A,p_D) is a local Nash equilibrium. To overcome this we use a Partially Input Convex Neural Network (PICNN) architecture [24] to obtain a convex relaxation of $-U_A,-U_D$, say $-\hat{U}_A,-\hat{U}_D$, respectively.

Our neural networks have constraints on the parameters, such that the output of the network is a convex function of a subset of the input sets. Motivated by the PICNN architecture proposed in [24], we present a modified architecture of partially input convex neural networks. This model defines a neural network with k layers, over the input y by implementing the following architecture, for $i=0,\ldots,k-1$,

$$z_{i+1}=g_i\Big(W_i^{(z)}z_i+W_i^{(y)}y+W_i^{(u)}u_i+b_i\Big),f(y;\theta)=z_k,u_0=x. \tag{10}$$

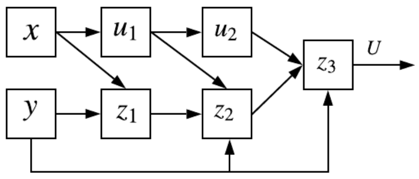

Fig. 1: Architecture of a partially input convex neural network with output U convex in input y but not necessarily convex in input x.

Here, u_i, z_i denote the hidden units for the "x-path" and "y-path", respectively, $\theta = \{W^{(y)}_{0:k-1}, W^{(u)}_{0:k-1}, W^{(z)}_{1:k-1}, b_{0:k-1}\}$ are the learnable parameters, and g_i represent the non-linear activation functions. The non-negative constraint on the $W^{(z)}$ terms can restrict the representation power of the neural network. In order to overcome this, $W^{(y)}$ terms are added to the network since they can be negative and thus endow the network with greater representation power. The convexity of the network is established below.

Proposition 1. *The function f is convex in y provided that all $W^{(z)}_{(1:k-1)}$ are non negative, and all functions g_i are convex and non-decreasing.*

Proof. The proof follows from the fact that the composition of a convex and convex non-decreasing function is convex and that non-negative sums of convex functions are also convex. For a given input pair (x, y), with fixed x, the term $W^{(u)}_i u_i$ in Eq. (10) is a constant. Note that, $W^{(y)}_i y$ is linear in y. Hence the output of the neural network is convex in y. $\square$

Figure 1 shows the architecture of a PICNN which takes as input (x, y) and outputs U such that U is convex in y and not necessarily convex in x. The neural network $\mathcal{F}_A$ is a PICNN with $x = p_D, y = p_A$ and $U = -\hat{U}_A$. Similarly, $\mathcal{F}_D$ is a PICNN with $x = p_A, y = p_D$ and $U = -\hat{U}_D$. Note that, the PICNN architecture contains "pass-through" layers that connect the input directly to the hidden layers so that the neural network has substantial representation power, which is otherwise restricted by the constraint on the weights [24].

The pseudo-code of the proposed approach is given in Algorithm V.1. We first generate random samples of strategies (p_A, p_D) (Step 1) and compute the corresponding U_A, U_D (Step 3). Then we train two PICNNs $\mathcal{F}_A$ and $\mathcal{F}_D$ to predict $\hat{U}_A$ and $\hat{U}_D$ (Step 4), respectively, using the data generated in Steps 1 and 2. Note that after training $\mathcal{F}_A$ and $\mathcal{F}_D$ can predict $-\hat{U}_A$, $-\hat{U}_D$, respectively, given an input pair (p_A, p_D) even when $\mathbb{P}$ is unknown.

To solve the game to find an approximate NE, we perform an alternating optimization by iteratively updating each player's policy in order to maximize its payoff. In every iteration, we first fix the defender's strategy and update the adversary's strategy using gradient descent, such that the adversary's payoff is maximized (Step 8). Then we, fix the adversary's strategy to the one obtained in Step 8 and update the defender's strategy, such that the defender's payoff is maximized (Step 10). The termination condition is attained when the individual change in the payoffs of players over two consecutive iterations is within a prespecified tolerance δ. The strategy pair $(\bar{p}_A, \bar{p}_D)$ returned by the algorithm is an approximate NE of the game, where the approximation factor depends on the convex approximation of the payoff functions by PICNNs.

Algorithm V.1 Supervised learning-based approach to solve DIFT-game with unknown transition probabilities

```
Input: Information flow graph $\mathcal{G}$, false-negatives $\mathbb{P}$ for all $v_i \in V_{\mathcal{G}}$, destination $d$, entry-point $\lambda$, $\delta > 0$, $\mu \in (0,1]$
Output: Approximate NE $(\bar{p}_A, \bar{p}_D)$
Data generation
 1: Generate random samples of $(p_A, p_D)$
 2: Compute $p_R^{(T)}(d), p_\tau^{(T)}, p_\phi^{(T)}$ using $\mathbb{P}$ and Eqs. (8) and (9)
 3: Compute $-U_A, -U_D$ using Eqs. (6) and (7), resp.
 4: Train $\mathcal{F}_A, \mathcal{F}_D$ using the data set from Steps 2 and 3
 5: Initialize randomly $\hat{p}_A^{(0)}$, $\hat{p}_D^{(0)}$, $k \leftarrow 0$, $-\hat{U}_A^{(0)} \leftarrow \mathcal{F}_A(\hat{p}_A^{(0)}, \hat{p}_D^{(0)})$, $-\hat{U}_D^{(0)} \leftarrow \mathcal{F}_D(\hat{p}_A^{(0)}, \hat{p}_D^{(0)})$, $\gamma > \delta$
Training
 6: while $\gamma > \delta$ do
 7:     Fix $\hat{p}_D$ and update $\hat{p}_A$ to minimize $-\hat{U}_A$ as: $\hat{p}_A^{(k+1)} \leftarrow \hat{p}_A^{(k)} - \mu \nabla_{p_A} \hat{U}_A^{(k)}$
 8:     Compute $-\hat{U}_A^{(k+1)} \leftarrow \mathcal{F}_A(\hat{p}_A^{(k+1)}, \hat{p}_D^{(k)})$
 9:     Fix $\hat{p}_A$ and update $\hat{p}_D$ to minimize $-\hat{U}_D$ as: $\hat{p}_D^{(k+1)} \leftarrow \hat{p}_D^{(k)} - \mu \nabla_{p_D} \hat{U}_D^{(k)}$
10:     Compute $-\hat{U}_D^{(k+1)} \leftarrow \mathcal{F}_D(\hat{p}_A^{(k+1)}, \hat{p}_D^{(k+1)})$
11:     $\gamma \leftarrow \max\left\{|\hat{U}_A^{(k+1)} - \hat{U}_A^{(k)}|, |\hat{U}_D^{(k+1)}, \hat{U}_D^{(k)}|\right\}$
12:     $k \leftarrow k+1$
13: end while
14: $\bar{p}_A \leftarrow \hat{p}_A^{(k)}$, $\bar{p}_D \leftarrow \hat{p}_D^{(k)}$
```

VI. Numerical Study

To demonstrate the performance of Algorithm V.1, we conduct 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) [3]. For both (i) and (ii) we show the convergence of Algorithm V.1 and validate that the convergent strategies are approximate NE through a sensitivity analysis. In all our experiments, we use the following values for parameters in Algorithm V.1: α_A = -500, α_D = 500 β_A = 500, β_D = -500, σ_A = -400 and σ_D = 400. During training we chose the learning rates of PICNNs to be $\eta_A = 0.05$ for $\mathcal{F}_A$ and $\eta_D = 0.2$ for $\mathcal{F}_D$ with a decay of factor 2 after every 10 epochs. The players' strategies are randomly initialized for the alternating optimization and step-size $\mu = 0.01$ and tolerance error level $\delta = 10^{-3}$.

Random Graphs: We generate a random DAG with 50 nodes. An edge exists in the graph with probability $(1+\delta)\log(n)/n$, where $\delta > 0$. Note that $\mathcal{G}$ for this construction is an Erdős-Renyi graph with the directed edge probability $(1+\delta)\log(n)/n$ [25]. The set of entry-points are randomly chosen as $\lambda = \{v_{24}, v_{38}, v_{32}\}$ and node v_{50} represents the

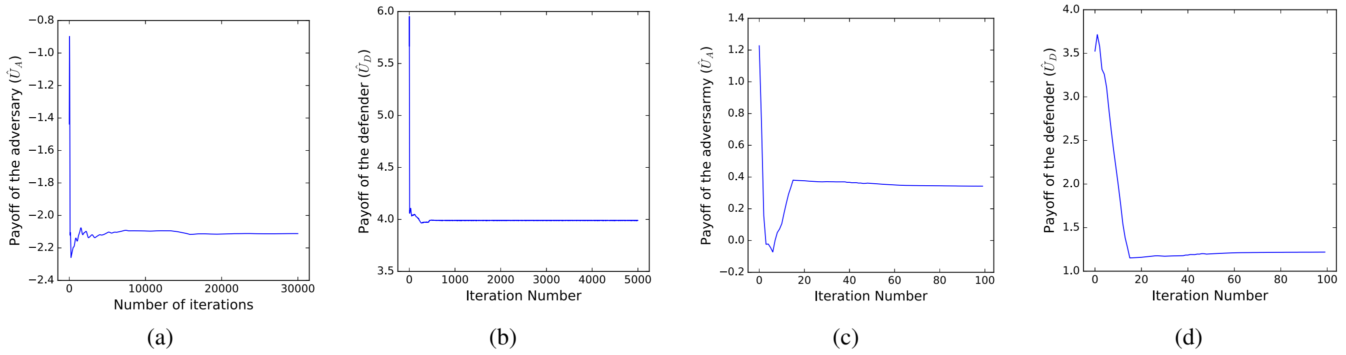

Fig. 2: Convergence experiments of Algorithm V.1 using PICNNs $\mathcal{F}_A$ and $\mathcal{F}_D$: (a) adversary's strategy for random graph, (b) defender's strategy for random graph, (c) adversary's strategy for ScreenGrab attack, and (d) defender's strategy ScreenGrab attack.

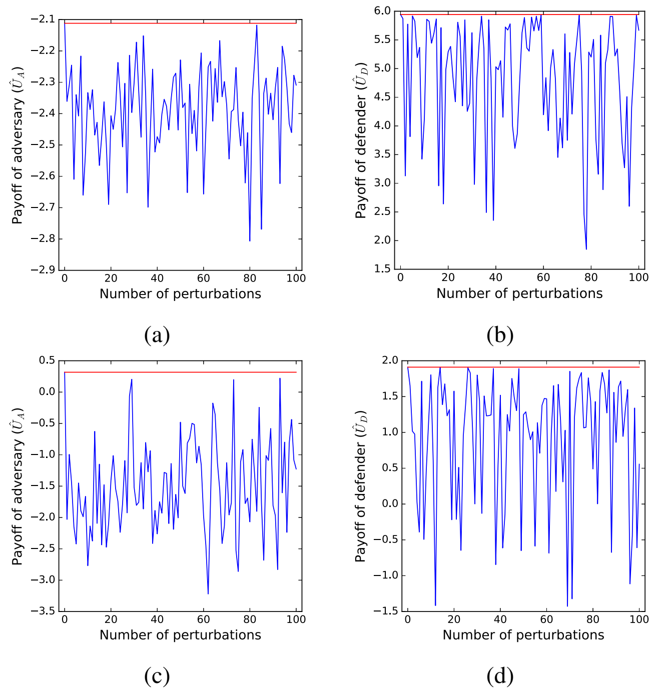

Fig. 3: Sensitivity results of the players for Case 1: (a) variation in the payoff of adversary w.r.t perturbed strategy for random graph, (b) variation in the payoff of defender w.r.t perturbed strategy for random graph, (c) variation in the payoff of adversary w.r.t perturbed strategy for ScreenGrab attack, and (d) variation in the payoff of adversary w.r.t perturbed strategy for ScreenGrab attack. The red lines in the plots indicate the payoff obtained from Algorithm V.1.

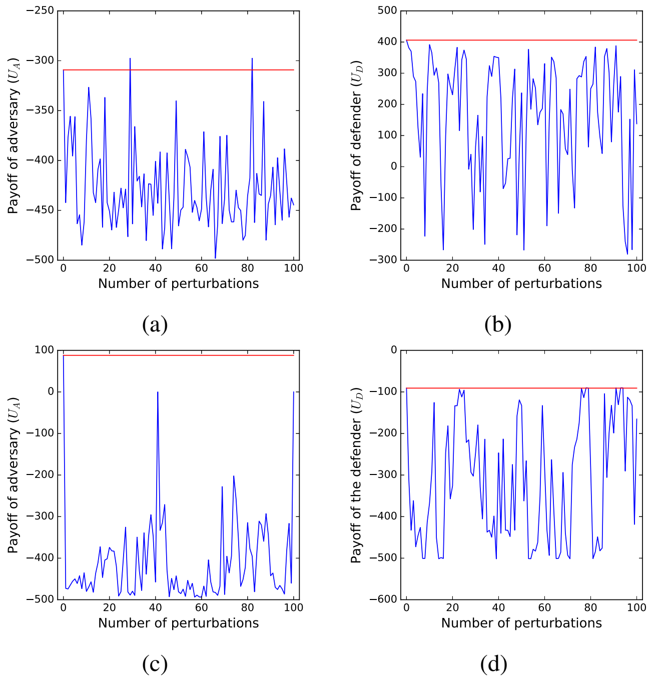

Fig. 4: Sensitivity results of the players for Case 2: (a) variation in the payoff of adversary w.r.t perturbed strategy for random graph, (b) variation in the payoff of defender w.r.t perturbed strategy for random graph, (c) variation in the payoff of adversary w.r.t perturbed strategy for ScreenGrab attack, and (d) variation in the payoff of adversary w.r.t perturbed strategy for ScreenGrab attack. The red lines in the plots indicate the payoff obtained from Algorithm V.1.

destination node d. The resource cost is randomly generated from a uniform distribution $[0,10]$.

ScreeGrab Attack: In a ScreenGrab attack, the adversary attempts to access the ScreenGrab to capture a screenshot of the victim's desktop and send it to the attacker's server. The information flow graph is built from system log collected using the RAIN system [3]. $\mathcal{G}$ consists of 12 nodes with $\lambda = v_1$, and $d = v_{12}$ denotes the ScreenGrab process which the adversary wants to gain access to. The resource cost is randomly generated from a uniform distribution $[0,10]$.

The convergence results of the players' strategies for random graph is given in Figure 2 (a), (b) and for the ScreenGrab attack data is given in Figure 2 (c), (d). We conduct a sensitivity analysis to validate that the converged strategies correspond to a NE of the game. We first initialize the strategies of both players to the strategy returned by Algorithm V.1. 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 V.1 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. We consider the following two cases for the sensitivity analysis.

Case 1: Perturb p_A (p_D, resp.) and compute the payoff $\hat{U}_A$ ($\hat{U}_D$, resp.) using the PICNN. Since $-\hat{U}_A$ and $-\hat{U}_D$ are convex approximated functions by construction, the payoff will not improve which is validated in Figures 3 (a)-(d).

Case 2: Perturb p_A (p_D, resp.) and compute the payoff U_A (U_D, resp.) using function evaluations of p_R, p_τ and p_ϕ as in Eqs. (8) and (9). Note that as the payoff functions $-U_A$ and $-U_D$ are non-convex in the strategies, the perturbation might result in an improved payoff of the player. The deviation from being a NE of the original game depends on the closeness of the convex approximated payoff functions, $\hat{U}_A, \hat{U}_D$, and the original payoff functions, U_A, U_D. Therefore, in this case we observe how well the supervised learning approach approximates the actual non-convex game formulation.

Figure 4 shows the payoffs corresponding to the perturbed strategies. As can be seen, for most of the random perturbations, the payoff obtained using Eqs. (8) and (9) 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.

VII. Conclusion

In this paper, we model the interaction of DIFT and adversarial information flows in a system with partial knowledge. We formulate the interaction between APTs and DIFT as a two-player, multi-stage stochastic game. The game model incorporates the probability with which the defender fails to detect an adversary despite trapping (false negatives). We first proved that the game terminates in finite steps and there exists a Nash equilibrium. To the best of our knowledge, there are no known solution approaches for nonzero-sum dynamic games with partial system information. To this end, we present a supervised learning-based algorithm to learn an approximate Nash equilibrium for the game when the transition probabilities are unknown. Our approach utilizes a partially input convex neural network architecture and an alternating optimization to update the players' strategies.The performance of our approach is demonstrated by two simulation studies; a random graph and a real-world dataset of the ScreenGrab attack obtained using the RAIN framework. We empirically show convergence of our algorithm to an approximate Nash equilibrium by conducting sensitivity analysis of the results obtained from the algorithm. As future work, we plan to characterize the approximation factor of the proposed approach and analyze the trade-off between learning of the function and obtaining an equilibrium.

References

[1] E. Cole, *Advanced persistent threat: Understanding the danger and how to protect your organization*. Newnes, 2012.

[2] G. E. Suh, J. W. Lee, D. Zhang, and S. Devadas, "Secure program execution via dynamic information flow tracking," *ACM Sigplan Notices*, vol. 39, no. 11, pp. 85–96, 2004.

[3] 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," *ACM SIGSAC Conference on Computer and Communications Security*, pp. 377–390, 2017.

[4] Q. Zhu, A. Clark, R. Poovendran, and T. Başar, "Deceptive routing games," *IEEE Conference on Decision and Control*, pp. 2704–2711, 2012.

[5] K.-w. Lye and J. M. Wing, "Game strategies in network security," *International Journal of Information Security*, vol. 4, no. 1-2, pp. 71–86, 2005.

[6] R. Amir, "Stochastic games in economics and related fields: An overview," *Stochastic Games and Applications*, pp. 455–470, 2003.

[7] Q. Zhu and T. Başar, "Robust and resilient control design for cyber-physical systems with an application to power systems," *IEEE Decision and Control and European Control Conference*, pp. 4066–4071, 2011.

[8] T. Alpcan and T. Başar, "An intrusion detection game with limited observations," *International Symposium on Dynamic Games and Applications*, vol. 26, 2006.

[9] K. C. Nguyen, T. Alpcan, and T. Başar, "Stochastic games for security in networks with interdependent nodes," *International Conference on Game Theory for Networks*, pp. 697–703, 2009.

[10] Q. Zhu, H. Tembine, and T. Başar, "Network security configurations: A nonzero-sum stochastic game approach," *American Control Conference*, pp. 1059–1064, 2010.

[11] L. Huang and Q. Zhu, "Adaptive strategic cyber defense for advanced persistent threats in critical infrastructure networks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 46, no. 2, pp. 52–56, 2019.

[12] M. O. Sayin, H. Hosseini, R. Poovendran, and T. Başar, "A game theoretical framework for inter-process adversarial intervention detection," *International Conference on Decision and Game Theory for Security*, pp. 486–507, 2018.

[13] D. Sahabandu, B. Xiao, A. Clark, S. Lee, W. Lee, and R. Poovendran, "DIFT games: Dynamic information flow tracking games for advanced persistent threats," *IEEE Conference on Decision and Control*, pp. 1136–1143, 2018.

[14] S. Moothedath, D. Sahabandu, A. Clark, S. Lee, W. Lee, and R. Poovendran, "Multi-stage dynamic information flow tracking game," *Conference on Decision and Game Theory for Security, Lecture Notes in Computer Science*, vol. 11199, pp. 80–101, 2018.

[15] D. Sahabandu, S. Moothedath, J. Allen, A. Clark, L. Bushnell, W. Lee, and R. Poovendran, "A game theoretic approach for dynamic information flow tracking with conditional branching," *American Control Conference*, 2019.

[16] K. Najim, A. S. Poznyak, and E. Gomez, "Adaptive policy for two finite markov chains zero-sum stochastic game with unknown transition matrices and average payoffs," *Automatica*, vol. 37, no. 7, pp. 1007–1018, 2001.

[17] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *Journal of machine learning research*, vol. 4, pp. 1039–1069, 2003.

[18] M. Bowling and M. Veloso, "Rational and convergent learning in stochastic games," *International joint conference on artificial intelligence*, vol. 17, no. 1, pp. 1021–1026, 2001.

[19] A. Greenwald, K. Hall, and R. Serrano, "Correlated Q-learning," *International Conference on Machine Learning*, vol. 3, pp. 242–249, 2003.

[20] M. N. Hossain, J. Wang, R. Sekar, and S. D. Stoller, "Dependence-preserving data compaction for scalable forensic analysis," *USENIX Security Symposium*, pp. 1723–1740, 2018.

[21] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Transactions on Computer Systems*, vol. 32, no. 2, p. 5, 2014.

[22] J. P. Hespanha and M. Prandini, "Nash equilibria in partial-information games on Markov chains," *IEEE Conference on Decision and Control*, vol. 3, pp. 2102–2107, 2001.

[23] A. B. Kahn, "Topological sorting of large networks," *Communications of the ACM*, vol. 5, no. 11, pp. 558–562, 1962.

[24] B. Amos, L. Xu, and J. Z. Kolter, "Input convex neural networks," *International Conference on Machine Learning*, vol. 70, pp. 146–155, 2017.

[25] P. Erdos and A. Rényi, "On the evolution of random graphs," *Institute of Mathematics, Hungarian Academy of Sciences*, vol. 5, no. 1, pp. 17–60, 1960.