Automatica 159 (2024) 111353

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Stochastic Dynamic Information Flow Tracking game using supervised learning for detecting advanced persistent threats☆

Shana Moothedath [a,*], Dinuka Sahabandu [b], Joey Allen [c], Linda Bushnell [b], Wenke Lee [c], Radha Poovendran [b]

[a] Department of Electrical and Computer Engineering, Iowa State University, USA
[b] Department of Electrical and Computer Engineering, University of Washington, USA
[c] College of Computing, Georgia Institute of Technology, USA

ARTICLE INFO

Article history:
Received 24 July 2020
Received in revised form 5 February 2022
Accepted 25 September 2023
Available online 30 October 2023

Keywords:
Cyber security
Stochastic games
Neural network
Advanced persistent threats
Dynamic information flow tracking

ABSTRACT

Advanced persistent threats (APTs) are organized prolonged cyberattacks by sophisticated attackers with the intent of stealing critical information. Although APT activities are stealthy and evade detection by traditional detection tools, they interact with the system components to make progress in the attack. These interactions lead to information flows that are recorded in the form of a system log. Dynamic Information Flow Tracking (DIFT) has been shown to be an effective way to detect APTs using information flows. A DIFT-based detection mechanism dynamically performs security analysis on the information flows to detect possible attacks. However, wide range security analysis using DIFT results in a significant increase in performance overhead and high rates of false-positives and false-negatives. In this paper, we model the strategic interaction between APT and DIFT as a non-cooperative stochastic game. The game unfolds on a state space constructed from an information flow graph (IFG) that is extracted from the system log. The objective of the APT in the game is to choose transitions in the IFG to find an optimal path in the IFG from an entry point of the attack to an attack target. On the other hand, the objective of DIFT is to dynamically select nodes in the IFG to perform security analysis for detecting APT. Our game model has imperfect information as the players are unaware of the actions of the opponent. We consider two scenarios of the game (i) the false-positive and false-negative rates of DIFT (i.e., transition probabilities of the game) are known and (ii) the false-positive and false-negative rates are unknown. For case (i), we propose a value iteration-based algorithm and prove that the solution converges to the optimal solution (Nash equilibrium). Case (ii) translates to an incomplete information game with unknown transition probabilities. For case (ii), we propose a supervised learning-based algorithm, referred to as Hierarchical Supervised Learning (HSL) algorithm. HSL integrates a neural network, to predict the value vector of the game, with a policy iteration algorithm to compute an approximate equilibrium. We implemented our algorithms for cases (i) and (ii) on real attack datasets for nation state and ransomware attacks and validated the performance of our approach. We compared the performance of the HSL algorithm when the transition probabilities are unknown with instances with known transition probabilities and demonstrated that HSL algorithm converges to a solution close to optimal (i.e., optimal value vector) while the value vector obtained using greedy does not converge to optimal for 44.4% of the states and the mean absolute error is almost 200 times that of the HSL.

1. Introduction

Advanced persistent threats (APTs) are sophisticated and prolonged cyberattacks that target specific high value organizations in sectors such as national defense, manufacturing, and the financial industry (Jang-Jaccard & Nepal, 2014; Watkins, 2014). APTs use advanced attack methods, including exploits of zero-day vulnerabilities, as well as highly-targeted spear phishing and other social engineering techniques to gain access to a network and then remain undetected for a prolonged period of time. The

☆ This work was supported by ONR grant N00014-16-1-2710 P00002, DARPA grant FA8650-15-C-7556, and NSF grant 2229876 and was supported in part by funds provided by the National Science Foundation (NSF), by the Department of Homeland Security, and by IBM. The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Claudio De Persis under the direction of Editor Christos G. Cassandras.

* Corresponding author.
E-mail addresses: mshana@iastate.edu (S. Moothedath), sdinuka@uw.edu (D. Sahabandu), jallen309@gatech.edu (J. Allen), lb2@uw.edu (L. Bushnell), wenke@cc.gatech.edu (W. Lee), rp3@uw.edu (R. Poovendran).

https://doi.org/10.1016/j.automatica.2023.111353
0005-1098/

attacking strategies of APTs are stealthy and are methodically designed to bypass conventional security mechanisms to cause more permanent, significant, and irreversible damages to the network.

Interaction of APTs with the network introduce information flows in the form of control flow and command flow commands which get recorded in the system log. Analyzing information flows is thus a promising approach to detect presence of APTs (Ji et al., 2017). Dynamic information flow tracking (DIFT) (Suh, Lee, Zhang, & Devadas, 2004) is a widely used defense mechanism to detect APTs. DIFT tags information flows originating from suspicious input channels in the network and tracks the propagation of the tagged flows. DIFT then initiates security check points, referred to as *traps*, to analyze the tagged flows and verify the authenticity of the flows. Since the ratio of the malicious flows to the benign flows is very small in the network, implementation of DIFT introduces significant performance overhead on the network and generates false-positives and false-negatives (Ji et al., 2017). Thus there is a need to selectively choose the security check points of DIFT.

This paper models detection of APTs using a DIFT-based detection mechanism. The effectiveness of detection depends on both the security policy of the DIFT and also on the actions of the APT. This strategic interaction motivates a game theoretic modeling as it allows us to investigate the trade-off between detection probability of the attacker and false-positive and false-negative rates of DIFT. We model the strategic interaction between APTs and DIFT as a stochastic game. The game unfolds on a state space that is defined using the information flow graph (IFG) constructed from the system log. The objective of the DIFT is to select locations in the system, i.e., nodes in the IFG, to place the security check points while minimizing the errors due to generation of false-positives and false-negatives. On the other hand, the objective of the APT is to choose an attack path starting from an entry point to a target node in the IFG. We note that, while APTs aim to achieve a reachability objective, the aim of DIFT is an optimal node selection. The APT-DIFT game has imperfect information structure as APTs are unaware of the locations at which DIFT places security check points and DIFT is unaware of the path chosen by the APT for the attack. We consider two scenarios of the game: (i) when the false-positive and false-negative rates are known to both players and (ii) when the false-positive and false-negative rates are unknown to both players. While case (i) is a complete information game, case (ii) is an incomplete information game with unknown transition probabilities.

We make the following contributions in this paper:

- We formulate a constant-sum stochastic game with total reward structure that models the interaction between the DIFT and APT. The game captures the information asymmetry between the attacker and the defender, false-positives, and false-negatives generated by DIFT.
- When the transition probabilities are known, we present a value iteration algorithm to compute equilibrium strategy and prove convergence and polynomial-time complexity.
- When the transition probabilities are unknown, we propose HSL (Hierarchical Supervised Learning), a supervised learning-based algorithm to compute an approximate equilibrium strategy of the incomplete information game. HSL integrates a neural network with policy iteration and utilizes the hierarchical structure of the state space of the APT-DIFT game to guarantee convergence.
- We implement our algorithms on attack data, of a nation state attack and a ransomware attack, collected using Refinable Attack INvestigation (RAIN) system, and show the convergence of the algorithm.

This paper is organized is as follows: Section 2 summarizes the related work. Section 3 elaborates on information flow graph, and the attacker and defender models. Section 4 presents the formulation of the stochastic game between APT and DIFT. Section 5 discusses the min–max solution concept used for solving the game. Section 6 presents the value iteration algorithm for the model-based approach and the supervised learning-based approach for the model-free case. Section 7 illustrates the results and discussions on the experiments conducted. Section 8 concludes the paper.

2. Related work

Stochastic games model the strategic interactions among multiple agents or players in dynamical systems and are well studied in the context of security games (Lye & Wing, 2005), economic games (Amir, 2003), and resilience of cyber–physical systems (Zhu & Başar, 2011). While stochastic games provide a strong framework for modeling security problems, often solving stochastic games are hard. There exists dynamic programming-based approaches, value-iteration and policy iteration, for computing Nash equilibrium (NE) strategies of these games. However, in many stochastic game formulations dynamic programming-based approaches do not converge. Refs. Ahmadi et al. (2018) and Ahmadi, Viswanathan, Ingham, Tan, and Ames (2020) modeled and studied the interaction between defender and attacker in a cyber setting as a two-player partially observable stochastic game and presented an approach for synthesizing sub-optimal strategies.

Multi-agent reinforcement learning (MARL) algorithms have been proposed in the literature to obtain NE strategies of zero-sum and nonzero-sum stochastic games, when the game information such as transition probabilities and payoff functions of the players are unknown. However, algorithms with guaranteed convergence are available only for special cases (Hu & Wellman, 1998, 2003; Najim, Poznyak, & Gomez, 2001; Prasad, Prashanth, & Bhatnagar, 2015). A Nash-Q learning algorithm is given in Hu and Wellman (2003) that converges to NE of a general-sum game when the NE is unique. The algorithm in Hu and Wellman (2003) is guaranteed to converge for discounted games with perfect information and unique NE. A multiagent Q-learning algorithm is presented in Hu and Wellman (1998) for discounted, constant-sum games with unknown transition probabilities when the players have perfect information. An adaptive policy approach using a regularized Lagrange function is proposed in Najim et al. (2001) for zero-sum games with unknown transition probabilities and irreducible state space. A stochastic approximation-based algorithm is presented in Prasad et al. (2015) for computing NE of discounted general-sum games with irreducible state space. While there exist algorithms that converge to NE of the game for special cases, there are no known algorithms to compute NE of a general stochastic game with incomplete information structure (Hu & Wellman, 1998; Prasad et al., 2015).

Detection of APTs are analyzed using game theory in the literature by modeling the interactions between APT and the detection mechanism as a game (Rass, König, & Schauer, 2020). A dynamic game model is given in Huang and Zhu (2019) to detect APTs that can adopt adversarial deceptions. A honeypot game theoretic model is introduced in Tian et al. (2019) to address detection of APTs.

There has been some effort to model the detection of APTs by a DIFT-based detection mechanism using game-theoretic framework (Moothedath et al., 2020a, 2020b; Sahabandu et al., 2019c). Ref. Moothedath et al. (2020a) studied a DIFT model which selects the trap locations in a dynamic manner and proposed a min-cut based solution approach. Detection of APTs when the attack

consists of multiple attackers, possibly with different capabilities, is studied in Sahabandu et al. (2019c). We note that, the game models in Moothedath et al. (2020a) and Sahabandu et al. (2019c) did not consider false-negative and false-positive generation by DIFT and are hence non-stochastic. Recently, stochastic models of APT-DIFT games was proposed in Moothedath et al. (2020b) and Sahabandu et al. (2019b). Ref. Sahabandu et al. (2019b) proposed a value iteration-based algorithm to obtain an ε-NE of the *discounted* game and Moothedath et al. (2020b) proposed a min-cut solution approach to compute an NE of the game. Formulations in Moothedath et al. (2020b) and Sahabandu et al. (2019b) assume that the transition probabilities of the game (false-positives and false-negatives) are known.

DIFT-APT games with unknown transition probabilities are studied in Sahabandu et al. (2019a, 2020). While Ref. Sahabandu et al. (2019a) proposed a two-time scale algorithm to compute an equilibrium of the discounted game, Sahabandu et al. (2020) considered an average reward payoff structure. Moreover, the game models in Sahabandu et al. (2019a, 2020) resulted in a unichain structure on the state space of the game, unlike the game model considered in this paper. The unichain structure of the game is critically utilized in Sahabandu et al. (2019a, 2020) for developing the solution approach and deriving the results. Ref. Misra et al. (2019), the preliminary conference version of this work, studied DIFT-APT game with unknown transition probabilities and average reward payoff structure, when the state space graph is not a unichain structure. The approach in Misra et al. (2019) approximated the payoff function to a convex function and utilized an input convex neural network (ICNN) architecture. The ICNN is integrated with an alternating optimization technique and empirical results are presented in Misra et al. (2019) to learn approximate equilibrium strategies. In this paper, we do not restrict the payoff function to be convex and hence relax the condition for the neural network to be input convex.

3. Preliminaries

3.1. Information Flow Graph (IFG)

Information Flow Graph (IFG) provides a graphical representation of the whole-system execution during the entire period of monitoring. We use the RAIN recording system (Ji et al., 2017) to construct the IFG. RAIN comprises a kernel module which logs all the system calls that are requested by the user-level processes in the target host. The target host then sends the recorded system recorded logs to the analysis host. The analysis host consists of a provenance graph builder, which then constructs the coarse-grained IFG. The coarse-IFG is then refined using various pruning techniques. A brief discussion on the pruning steps is included in Section 7. For more details, please refer Ji et al. (2017).

IFGs are widely used by analysts for effective cyber response (Hossain, Wang, Sekar, & Stoller, 2018; Ji et al., 2017). IFGs are directed graphs, where the nodes in the graph form entities such as 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 (Hossain et al., 2018). 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 corase-grained-IFG which is then pruned and refined incrementally to obtain the corresponding IFG (Ji et al., 2017).

3.2. Attacker model

This paper consider advanced cyberattacks called as APTs. APTs enter into the system by leveraging vulnerabilities in the system and implement multiple sophisticated methods to continuously and stealthily steal information (Hossain et al., 2018). A typical APT attack consists of multiple stages initiated by a successful penetration and followed by initial compromise, C&C communications, privilege escalation, internal reconnaissance, exfiltration, and cleanup (Hossain et al., 2018). Detection of APT attacks are very challenging as the attacker activities blend in seamlessly with normal system operation. Moreover, as APT attacks are customized, they cannot be detected using signature-based detection methods such as firewalls, intrusion detection systems, and antivirus software. Several different approaches have been proposed for detecting APTs. The authors of Vance (2014) proposed a statistical anomaly detection approach to analyze network based communications in order to detect APTs. While Lajevardi and Amini (2019) utilized the correlations between operating system and network events and developed a semantic-based approach for detecting APTs, Siddiqui, Khan, Ferens, and Kinsner (2016) proposed a fractal based anomaly classification mechanism. A game-theoretic approach to model a long-term interaction between a stealthy attacker and a proactive defender was proposed in Huang and Zhu (2020). In this work, we utilize the information flows generated in the system to model the interactions of the APT and consider dynamic information flow tracking-based defense mechanism, which is discussed in detail below.

3.3. Defender model

Dynamic information flow tracking (DIFT) is a promising technique to detect security attacks on computer systems (Clause, Li, & Orso, 2007; Enck et al., 2014; Suh et al., 2004). DIFT tracks the system calls to detect the malicious information flows from an adversary and to restrict the use of these malicious flows. DIFT architecture is composed of (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 suspicious 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 in the system, DIFT initiates security analysis and tagged flows are inspected by DIFT at specific locations, referred to as *traps*. DIFT conducts fine grain analysis at traps to detect the attack and to perform risk assessment. While tagging and trapping using DIFT is a promising detection mechanism against APTs, DIFT introduces performance overhead on the system. Performing security analysis (trapping) of tagged flows uses considerable amount of memory of the system (Enck et al., 2014). Thus there is a tradeoff between system log granularity and performance (Ji et al., 2017).

4. Problem formulation

This section formulates the interaction between the APT and the DIFT-based defense mechanism as a stochastic game where the decisions taken by the adversary (APT) and the defender (DIFT), referred to as agents/players, influences the system behavior. The objective of the adversary is to choose transitions in the IFG so as to reach the destination node set $\mathcal{D} \subset V_{\mathcal{G}}$ from the set of entry points $\lambda \subset V_{\mathcal{G}}$. On the other hand, the objective of the defender is to dynamically choose locations to trap the information flow so as to secure the system from any possible attack. The state of the system denotes the location of the tagged information flow in the system. The defender cannot distinguish a malicious and a benign flow. 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. We consider an intelligent adversary who is inside the system and can observe the system activities. Hence the attacker knows the positions of the flows in the system, and also which flows are tagged. Thus the state of the game is known unlike in a partially observable game setting where the state of the game is unknown. The system's current state and the joint actions of the agents together determine a probability distribution over the possible next states of the system.

4.1. State space

The state of the game at a time step t, denoted as s_t, is defined as the position of the tagged information flow in the system. Let $\mathbf{S} = \{v_0, v_1, \ldots, v_N, \phi, \tau_A, \tau_B\}$ be the finite state space of the game. Here, $s_t = \phi$ denotes the state when the tagged flow drops out by abandoning the attack, $s_t = \tau_A$ denotes the state when DIFT detects the adversary after performing the security analysis, and $s_t = \tau_B$ denotes the state when DIFT performs security analysis and concludes a benign flow as malicious (false positive). Let $\mathcal{D}$ be the destination (target) node set of the adversary and $|\mathcal{D}| = q$. Without loss of generality, let $\mathcal{D} = \{v_1, v_2, \ldots, v_q\}$. We assume that both agents know the IFG and the destination set $\mathcal{D}$. The state v_0 corresponds to a virtual state that denote the starting point of the game. In the state space $\mathbf{S}$, v_0 is connected to all nodes in λ. Thus the state of the game at $t = 0$ is $s_0 = v_0$.

4.2. Action spaces

At every time instant in the game, the players choose their actions from their respective action sets. The action set of the adversary is defined as the possible set of transitions the adversarial flow can execute at a state. The defender has limited resources and hence can perform security analysis only one information flow at a time. Thus the defender's action set is defined such that, at every decision point, the defender chooses one node to perform security analysis, among the possible nodes that the adversary can transition to.

Let the action set of the adversary and the defender at a state s_t be $\mathcal{A}_A(s_t)$ and $\mathcal{A}_D(s_t)$, respectively. At a state $s_t \in V_{\mathcal{G}}$, the adversary chooses an action either to drop out, i.e., abort the attack, or to continue the attack by transitioning to a neighboring node. Thus $\mathcal{A}_A(s_t) := \{\phi\} \cup \{v_j : s_t = v_i \text{ and } (v_i, v_j) \in E_{\mathcal{G}}\}$. On the other hand, the defender's action set at state s_t $\mathcal{A}_D(s_t) := \{0\} \cup \{v_j : s_t = v_i \text{ and } (v_i, v_j) \in E_{\mathcal{G}}\}$, where $\mathcal{A}_D(s_t) = 0$ represents that the tagged information flow is not trapped and $\mathcal{A}_D(s_t) = v_j$ represents that defender decides to perform security analysis at the node $v_j \in V_{\mathcal{G}}$ at instant t.

The game originates at $t = 0$ with state $s_0 = v_0$ with $\mathcal{A}_A(v_0) := \lambda$ and $\mathcal{A}_D(v_0) := 0$. The definition of action sets at v_0 captures the fact that the adversarial flow originates at one of the entry points. Performing security analysis at entry points cannot detect the attack as there are not enough traces to analyze. Moreover, the game terminates at time t if the state of the game $s_t \in \{\phi, \tau_A, \tau_B\} \cup \mathcal{D}$. The set of states $\{\phi, \tau_A, \tau_B\} \cup \mathcal{D}$ are referred to as the *absorbing states* of the game. For $s_t \in \{\phi, \tau_A, \tau_B\} \cup \mathcal{D}$, $\mathcal{A}_D(s_t) = \mathcal{A}_A(s_t) = \emptyset$. At a non-absorbing state, the players choose their actions from their respective action set and the game evolves until the state of the game is an absorbing state.

4.3. Transitions

The transition probabilities of the game are governed by the uncertainty associated with DIFT. DIFT is not capable of performing security analysis accurately due to the generation of false-positives and false-negatives. Consider a tagged flow incoming at node $v_i \in V_{\mathcal{G}}$ at time t. Let the action chosen by the defender and the adversary at s_t be d_t and a_t, respectively. If defender chooses not to trap an information flow, then the flow proceeds to the node in $\mathcal{G}$ chosen by the adversary. That is, if $d_t = 0$, then $s_{t+1} = a_t$. If the defender chooses to trap the node at which the adversary also decides to transition to, then the adversary is detected with probability $1 - FN(d_t)$ and the flow transition to the node in $\mathcal{G}$ corresponding to the action of the adversary with the remaining probability. That is, if $d_t = a_t = v_i$, then $s_{t+1} = \tau_A$ with probability $1 - FN(d_t)$ and $s_{t+1} = a_t$ with probability $FN(d_t)$. If the defender decides to trap a node which is not the node the adversary decides to transition to, then the defender generates a false-positive with probability $FP(d_t)$ and the flow transition to the node in $\mathcal{G}$ corresponding to the action of the adversary with the remaining probability. That is, if $d_t \neq 0$ and $d_t \neq a_t$, then $s_{t+1} = \tau_B$ with probability $FP(d_t)$ and $s_{t+1} = a_t$ with probability $1 - FP(d_t)$.

Let $P(s_t, s_t, d_t, s_{t+1})$ denote the probability of transitioning to a state s_{t+1} from a state s_t under actions a_t and d_t. Then,

$$P(s_t, a_t, d_t, s_{t+1}) = \begin{cases} 1, & s_{t+1} = a_t, \quad \text{if } d_t = 0 \\ FN(d_t), & s_{t+1} = a_t, \quad \text{if } d_t = a_t \\ 1 - FN(d_t), & s_{t+1} = \tau_A, \quad \text{if } d_t = a_t \\ FP(d_t), & s_{t+1} = \tau_B, \quad \text{if } d_t \neq a_t \\ 1 - FP(d_t), & s_{t+1} = a_t, \quad \text{if } d_t \neq a_t \end{cases} \tag{1}$$

The transition probabilities $FN(v_i)$, $FP(v_i)$, for $v_i \in V_{\mathcal{G}}$, denote the rates of generation of false-negatives and false-positives, respectively, at the different nodes in the IFG. We note that, different nodes in IFG have different capabilities to perform security analysis and depending on that the value of $FN(\cdot)$ and $FP(\cdot)$ are different. The numerical values of these transition probabilities also depend on the type of the attack. As APTs are tailored attacks that can manipulate the system operation and evade conventional security mechanisms such as firewalls, anti-virus software, and intrusion-detection systems, $FN(\cdot)$'s and $FP(\cdot)$'s are often unknown and hard to estimate accurately.

4.4. Strategies

A strategy for a player is a mapping which yields probability distribution over the player's actions at every state. Consider *mixed* (stochastic) and stationary player strategies. When the strategy is stationary, the probability of choosing an action at a state depends only on the current state of the game. Let the stationary strategy space of the attacker be $\mathbf{P}_A$ and that of the defender be $\mathbf{P}_D$. 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 $p_D \in \mathbf{P}_D$. Here, $p_D = [p_D(v_{q+1}), \ldots, p_D(v_N)]$, where $p_D(v_i)$ denotes the probability distribution over all the actions of the defender at state v_i, i.e., out-neighboring nodes of v_i in IFG and not trapping. For $p_A = [p_A(v_0), p_A(v_{q+1}), \ldots, p_A(v_N)]$, $p_A(v_i)$ is a probability distribution over all possible out-neighbors of the node v_i in the IFG and ϕ. We note that, $\mathcal{A}_A(s_t) = \mathcal{A}_D(s_t) = \emptyset$, for $s_t \in \{\phi, \tau_A, \tau_B\} \cup \mathcal{D}$. Also, $\mathcal{A}_D(v_0) = \emptyset$.

4.5. Payoffs

In the APT-DIFT game, the aim of the APT is to choose transitions in the IFG in order to reach destination by evading detection by DIFT. The aim of DIFT is to select the security check points to secure the system from the APT attack. Recall that APTs are stealthy attacks that undergo for a long period of time with the ultimate goal of stealing information over a long time. The destructive consequences of APTs are often unnoticeable until the final

stages of the attack (Bencsáth, Pék, Buttyán, & Felegyhazi, 2012). In this paper we consider the payoff functions of the APT-DIFT game such that players achieve reward (β) when their respective aim is achieved. Such a reward structure is used various classes of security games, including interdiction games (Zhang, Guo, An, Tran-Thanh, & Jennings, 2019), network security games (Wang, Perrault, Mate, & Tambe, 2020), and patrolling games (Vorobeychik, An, & Tambe, 2012). DIFT achieves the aim under two scenarios: (1) when the APT is detected successfully and (2) when APT drops out the attack. We note that, in both cases (1) and (2), DIFT secures the system from the APT attack and hence is rewarded β. On the other hand, APT achieves the aim under two scenarios: (i) when APT reach destination and (ii) when a benign flow is concluded as malicious, i.e., false-positive. We note that, when a benign flow is concluded as malicious, DIFT no longer analyzes the flows (as it believes APT is detected) and hence the actual malicious flow evades detection and can achieve the aim. Thus in both cases (i) and (ii) APT achieves the aim and receives a reward of β.

Let the payoff of player k at an absorbing state s_t be $r^k(s_t)$, where $k \in \{A, D\}$. At state τ_A DIFT receives a payoff of β and APT receives 0 payoff. At state τ_B the APT receives a payoff of β and DIFT receives 0 payoff. At a state in the set $\mathcal{D} = \{v_1, v_2, \ldots, v_q\}$, APT receives a payoff of β and DIFT receives 0 payoff. At state ϕ DIFT receives a payoff of β and APT receives 0 payoff. Then,

$$r^A(s_t) = \begin{cases} \beta, & s_t \in \mathcal{D} \\ \beta, & s_t = \tau_B \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$r^D(s_t) = \begin{cases} \beta, & s_t = \tau_A \\ \beta, & s_t = \phi \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

At each stage in game, s_t at time t, both players simultaneously choose their action a_t and d_t and transition to a next state s_{t+1}. This is continued until they reach an absorbing state and receive the payoff defined using Eqs. (2) and (3).

Let T denote termination time of the game, i.e., $s_{t+1} = s_t$ for $t \geqslant T$. At the termination time $s_T \in \{\phi, \tau_A, \tau_B\} \cup \mathcal{D}$. Let U_A and U_D denote the payoff functions of the adversary and the defender, respectively. As the initial state of the game is v_0, for a strategy pair (p_A, p_D) the expected payoffs of the players are

$$U_A(v_0, p_A, p_D) = \mathbb{E}_{v_0, p_A, p_D}\left[\sum_{t=0}^{T} r^A(s_t)\right] = \mathbb{E}_{v_0, p_A, p_D}\left[r^A(s_T)\right], \tag{4}$$

$$U_D(v_0, p_A, p_D) = \mathbb{E}_{v_0, p_A, p_D}\left[\sum_{t=0}^{T} r^D(s_t)\right] = \mathbb{E}_{v_0, p_A, p_D}\left[r^D(s_T)\right], \tag{5}$$

where $\mathbb{E}_{v_0, p_A, p_D}$ denotes the expectation with respect to p_A and p_D when the game originates at state v_0. The objective of APT and DIFT is to individually maximize their expected total payoff given in Eqs. (4) and (5), respectively. Thus the optimization problem solved by DIFT is

$$\max_{p_D \in \mathbf{P}_D} U_D(v_0, p_A, p_D). \tag{6}$$

Similarly, the optimization problem of the APT is

$$\max_{p_A \in \mathbf{P}_A} U_A(v_0, p_A, p_D). \tag{7}$$

To bring out the structure of the payoffs well, we let $R_{s_0}(s)$ be the cumulative probability with which the state of the game at the termination time is s, when the game originates at s_0. With slight abuse of notation, we use $R_{s_0}(\mathcal{D})$ to denote the cumulative probability with which the state of the game at the termination time lies in set $\mathcal{D}$, when the game originates at s_0. At the time of termination, i.e., $t = T$, the state of the game satisfies one of the following: (i) $s_T = \tau_A$, (ii) $s_T = \tau_B$, (iii) $s_T \in \mathcal{D}$, and (iv) $s_T = \phi$. Using these definitions Eqs. (4) and (5) can be rewritten as

$$U_A(v_0, p_A, p_D) = \Big(R_{s_0}(\mathcal{D}) + R_{s_0}(\tau_B)\Big)\,\beta, \tag{8}$$

$$U_D(v_0, p_A, p_D) = \Big(R_{s_0}(\tau_A) + R_{s_0}(\phi)\Big)\beta. \tag{9}$$

Using the reformulation of the payoff functions, we present the following property of the APT-DIFT game.

Proposition 4.1. *The APT-DIFT stochastic game is a constant sum game.*

Proof. Recall that APT-DIFT game has absorbing states ϕ, τ_A, τ_B and $\mathcal{D}$. At the termination time of the game, i.e., $t = T$, the state of the game $s_T \in \{\phi, \tau_A, \tau_B\} \cup \mathcal{D}$. This implies

$$R_{s_0}(\phi) + R_{s_0}(\tau_A) + R_{s_0}(\tau_B) + R_{s_0}(\mathcal{D}) = 1.$$

This gives $U_A(v_0, p_A, p_D) + U_D(v_0, p_A, p_D) = \Big(R_{s_0}(\phi) + R_{s_0}(\tau_A) + R_{s_0}(\tau_B) + R_{s_0}(\mathcal{D})\Big)\,\beta = \beta$. Thus the game between APT and DIFT is a constant sum game with constant equal to β. □

5. Solution concept

The solution concept of the game is defined as follows. Each player is interested in maximizing their individual reward in the *minimax* sense. In other words, each player is assuming the worst case for an optimal opponent player. Since the APT-DIFT game is a constant-sum game, it is sufficient to view that the opponent player is acting to minimize the reward of the agent. Thus DIFT chooses its actions to find an optimal strategy $p_D^\star$ that achieves the upper value defined as

$$\overline{V}(s_0) = \max_{p_D \in \mathbf{P}_D} \min_{p_A \in \mathbf{P}_A} U_D(v_0, p_A, p_D). \tag{10}$$

On the other hand, APT chooses its actions to find an optimal strategy $p_A^\star$ that achieves

$$\max_{p_A \in \mathbf{P}_A} \min_{p_D \in \mathbf{P}_D} U_A(v_0, p_A, p_D) = \max_{p_A \in \mathbf{P}_A} \min_{p_D \in \mathbf{P}_D} \Big(\beta - U_D(v_0, p_A, p_D)\Big). \tag{11}$$

This is equivalent to saying that APT aims to find an optimal strategy $p_A^\star$ that achieves the lower value defined as

$$\underline{V}(s_0) = \min_{p_A \in \mathbf{P}_A} \max_{p_D \in \mathbf{P}_D} U_D(v_0, p_A, p_D). \tag{12}$$

From Eqs. (10) and (12), the defender tries to maximize and the adversary tries to minimize $U_D(v_0, p_A, p_D)$. Hence the value of the game is defined as

$$\begin{aligned} V^\star(s_0) &:= \max_{p_D \in \mathbf{P}_D} U_D(v_0, p_A^\star, p_D) = U_D(v_0, p_A^\star, p_D^\star) \\ &= \min_{p_A \in \mathbf{P}_A} U_D(v_0, p_A, p_D^\star). \end{aligned} \tag{13}$$

The strategy pair $(p_A^\star, p_D^\star)$ is referred to as the saddle point or Nash equilibrium (NE) of the game.

Definition 5.1. Let $(p_A^\star, p_D^\star)$ be a Nash equilibrium of the APT-DIFT game. A strategy pair (p_A, p_D) is said to be an $\varepsilon-$Nash equilibrium, for $\varepsilon > 0$, if

$$U_D(v_0, p_A, p_D) \geqslant U_D(v_0, p_A^\star, p_D^\star) - \varepsilon.$$

In other words, the value corresponding to the strategy pair (p_A, p_D) denoted as $V(s_0)$ satisfies

$$V(s_0) \geqslant V^\star(s_0) - \varepsilon.$$

Proposition 5.2 proves the existence of NE for the APT-DIFT game.

Proposition 5.2. *There exists a Nash equilibrium for the APT-DIFT stochastic game.*

Proof of Proposition 5.2 is presented in the Appendix.

In the next section we present our approach to compute an NE of the APT-DIFT game.

6. Solution to the APT-DIFT game

This section presents our approach to compute an NE of the APT-DIFT game. Firstly, we propose a model-based approach to compute NE using a value iteration algorithm. Later, we present a model-free approach based on a policy iteration algorithm, when the transition probabilities, i.e., the rate of false-positives and false-negatives, are unknown.

6.1. Value iteration algorithm

This subsection presents our solution approach for solving the APT-DIFT game when the false-positive and false-negative rates (transition probabilities) are known. By Proposition 5.2, there exists an NE for the APT-DIFT game. Our approach to compute NE of the APT-DIFT game is presented below.

Let $s \in \mathbf{S}$ be an arbitrary state of the APT-DIFT game. The state value function for a constant-sum game, analogous to the Bellman equation, can be written as

$$V^\star(s) = \max_{p_D(s)\in\mathbf{P}_D(s)} \min_{a\in\mathcal{A}_A(s)} \sum_{d\in\mathcal{A}_D(s)} Q^\star(s,a,d)\, p_D(s,d), \tag{14}$$

where the Q-values are defined as

$$Q^\star(s,a,d) = \sum_{s'\in\mathbf{S}} P(s,a,d,s')V^\star(s'). \tag{15}$$

The min in Eq. (14) can also be defined over mixed (stochastic) policies, but, since it is 'inside' the max, the minimum is achieved for a deterministic action choice (Littman, 2001). The strategy selected by Eq. (14) is referred to as the *minimax* strategy, and given by

$$p_D^\star(s) = \arg\max_{p_D(s)\in\mathbf{P}_D(s)} \min_{a\in\mathcal{A}_A(s)} \sum_{d\in\mathcal{A}_D(s)} Q^\star(s,a,d)\, p_D(s,d). \tag{16}$$

The aim here is to compute minimax strategy $p_D^\star$. Our proposed algorithm and convergence proof is given below.

Let $V = \{V(s) : s \in \mathbf{S}\}$ denote the value vector corresponding to a strategy pair (p_A, p_D). The value for a state $s \in \mathbf{S}$, $V(s)$, is the expected payoff under strategy pair (p_A, p_D) if the game originates at state s. Then, $V^\star(s)$ is the expected payoff corresponding to an NE strategy pair $(p_A^\star, p_D^\star)$ if the game originates at state s. Algorithm 6.1 presents the pseudocode to compute the value vector of the APT-DIFT game.

Algorithm 6.1 is a value iteration algorithm defined on a value vector $V = \{V(s) : s \in \mathbf{S}\}$, where $V(s)$ is the value of the game starting at the state s. Thus $V(s) = \beta$, for $s \in \{\phi, \tau_A\}$, and $V(s') = 0$, for $s' \in \{\tau_B\} \cup \mathcal{D}$. The values of the states is computed recursively, for every iteration $k = 1, 2, \ldots$, $V^{(k)}(s)$, as follows:

$$V^{(k)}(s) = \begin{cases} \beta, & \text{if } s \in \{\phi, \tau_A\}, \\ 0, & \text{if } s \in \{\tau_B\} \cup \mathcal{D}, \\ \text{val}(s, V^{(k-1)}), & \text{otherwise,} \end{cases} \tag{17}$$

where $\text{val}(s, V^{(k-1)}) =$

$$\max_{p_D\in\mathbf{P}_D} \min_{a\in\mathcal{A}_A(s)} \sum_{s'\in\mathbf{S}} \sum_{d\in\mathcal{A}_D(s)} p_D(s,d)P(s,a,d,s')V^{(k-1)}(s').$$

Algorithm 6.1 Value iteration algorithm to find NE strategy in APT-DIFT game

Input: APT-DIFT game with state space $\mathbf{S}$, destination set $\mathcal{D}$, action space $\mathcal{A}_D$, $\mathcal{A}_A$, payoff parameter β, transition probabilities $FN(\cdot)$, $FP(\cdot)$

Output: Value vector $\hat{V}$ and defender policy $\hat{p}_D$

1: Initialize value vector $V^{(0)}(s) \leftarrow 0$, for all $s \in \mathbf{S}$ and $V^{(1)}(s') \leftarrow \beta$ for $s' \in \{\tau_A, \phi\}$, $V^{(1)}(s'') \leftarrow 0$ for all $s'' \in \mathbf{S} \setminus \{\tau_A, \phi\}$, $k \leftarrow 0$, $\varepsilon \geqslant 0$

2: **while** $\max\{|V^{(k+1)}(s) - V^{(k)}(s)| : s \in \mathbf{S}\} > \varepsilon$ **do**

3: $k \leftarrow k + 1$

4: **for** $s \notin \{\phi, \tau_A, \tau_B\} \cup \mathcal{D}$ **do**

5: $V^{(k+1)}(s) \leftarrow \max_{p_D\in\mathbf{P}_D} \min_{a\in\mathcal{A}_A(s)} \sum_{s'\in\mathbf{S}} \sum_{d\in\mathcal{A}_D(s)} p_D(s,d)P(s,a,d,s')V^{(k)}(s')$

6: **end for**

7: **end while**

8: **return** Vector $\hat{V}$, where $\hat{V}(s) \leftarrow V^{(k)}(s)$

9: Compute DIFT strategy $\hat{p}_D$, $\hat{p}_D(s) \leftarrow \arg\max_{p_D\in\mathbf{P}_D} \min_{a\in\mathcal{A}_A(s)} \sum_{s'\in\mathbf{S}} \sum_{d\in\mathcal{A}_D(s)} p_D(s,d)P(s,a,d,s')\hat{V}(s')$

The value-iteration algorithm computes a sequence $V^{(0)}$, $V^{(1)}$, $V^{(2)}, \ldots$, where for $k = 0, 1, 2 \ldots$, each valuation $V^{(k)}$ associates with each state $s \in \mathbf{S}$ a lower bound $V^{(k)}(s)$ on the value of the game. Algorithm 6.1 need not terminate in finitely many iterations (Royden & Fitzpatrick, 2010). The parameter ε, in step 2, denotes the acceptable error bound which is the maximum absolute difference in the values corresponding to two consecutive iterations, i.e., $\max\{|V^{(k)}(s) = V^{(k+1)}(s)| : s \in \mathbf{S}\}$, and serves as a stopping criteria for Algorithm 6.1. Smaller value of ε in Algorithm 6.1 will return a value vector that is closer to the actual value vector. Below we prove that as k approaches ∞, the values $V^{(k)}(s)$ approaches the exact values $V(s)$ from below, i.e., $\lim_{k\to\infty} V^{(k)}(s)$ converges to the value of the game at state s. Theorem 6.3 proves the asymptotic convergence of the values.

Lemma 6.1. *Consider the value iteration algorithm in Algorithm 6.1. Let $V^{(k+1)}$ and $V^{(k)}$ be the value vectors corresponding to iterations $k+1$ and k, respectively. Then, $V^{(k+1)}(s) \geqslant V^{(k)}(s)$, for all $s \in \mathbf{S}$.*

Proof. We first prove that result for a state $s \in \{\phi, \tau_A, \tau_B\} \cup \mathcal{D}$. From Eq. (17), for every iteration $k = 1, 2, \ldots$ $V^{(k)}(s) = \beta$, for $s \in \{\phi, \tau_A\}$, and $V^{(k)}(s) = 0$, for $s \in \{\tau_B\} \cup \mathcal{D}$. From the initialization step of Algorithm 6.1 (Step 1, $V^{(0)}(s) = 0$ for all $s \in \mathbf{S}$. Thus for any state s, where $s \in \{\phi, \tau_A, \tau_B\} \cup \mathcal{D}$, $V^{(k+1)}(s) \geqslant V^{(k)}(s)$ for any arbitrary iteration k of Algorithm 6.1.

For an arbitrary state s, where $s \in \mathbf{S} \setminus \{\phi, \tau_A, \tau_B\} \cup \mathcal{D}$, we prove the result using an induction argument. The induction hypothesis is that arbitrary iterations k and $k+1$ satisfy $V^{(k+1)}(s) \geqslant V^{(k)}(s)$, for all $s \in \mathbf{S} \setminus \{\phi, \tau_A, \tau_B\} \cup \mathcal{D}$.

Base step: Consider $k = 0$ as the base step. Initialize $V^{(0)}(s) = 0$ for all $s \in \mathbf{S} \setminus \{\phi, \tau_A, \tau_B\} \cup \mathcal{D}$ and set $V^{(1)}(s) = 0$ for all $s \in \mathbf{S} \setminus \{\phi, \tau_A, \tau_B\} \cup \mathcal{D}$. This gives $V^{(1)}(s) \geqslant V^{(0)}(s)$, for all $s \in \mathbf{S} \setminus \{\phi, \tau_A, \tau_B\} \cup \mathcal{D}$.

Induction step: For the induction step, assume that iteration k satisfies $V^{(k)}(s) \geqslant V^{(k-1)}(s)$ for all $s \in \mathbf{S} \setminus \{\phi, \tau_A, \tau_B\} \cup \mathcal{D}$.

Consider iteration $(k+1)$. Then,

$$\begin{aligned} V^{(k+1)}(s) &= \max_{p_D\in\mathbf{P}_D} \min_{a\in\mathcal{A}_A(s)} \sum_{s'\in\mathbf{S}} \sum_{d\in\mathcal{A}_D(s)} p_D(s,d)P(s,a,d,s')V^{(k)}(s') \\ &\geqslant \min_{a\in\mathcal{A}_A(s)} \sum_{s'\in\mathbf{S}} \sum_{d\in\mathcal{A}_D(s)} p_D^{(k)}(s,d)P(s,a,d,s')V^{(k)}(s') \end{aligned} \tag{18}$$

$$\geqslant \min_{a\in\mathcal{A}_A(s)} \sum_{s'\in\mathbf{S}} \sum_{d\in\mathcal{A}_D(s)} p_D^{(k)}(s,d)P(s,a,d,s')V^{(k-1)}(s') \quad (19)$$

$$= \max_{p_D\in\mathbf{P}_D} \min_{a\in\mathcal{A}_A(s)} \sum_{s'\in\mathbf{S}} \sum_{d\in\mathcal{A}_D(s)} p_D(s,d)P(s,a,d,s')V^{(k-1)}(s')$$

$$= V^{(k)}(s) \quad (20)$$

Eq. (18) holds as the value obtained using a maximizing policy is at least as the value obtained using a specific policy p_D. Eq. (19) follows from the induction argument and Eq. (20) is from the definition of $V^{(k)}(s)$. This completes the proof. □

Proposition 6.2 (*Monotone Convergence Theorem, Royden and Fitzpatrick (2010)*)**.** *If a sequence is monotone increasing and bounded from above, then it is a convergent sequence.*

The value of any state $s \in \mathbf{S}$ is bounded above by β. Using Lemma 6.1 and Proposition 6.2, we state the convergence result of Algorithm 6.1.

Theorem 6.3. *Consider the value iteration algorithm in Algorithm 6.1. Let $V^{(k)}(s)$, $V^\star(s)$ be the value at iteration k and the optimal value of state $s \in \mathbf{S}$, respectively. Then, as $k \to \infty$, $V^{(k)}(s) \to V^\star(s)$, for all $s \in \mathbf{S}$. Further, the output of Algorithm 6.1, $\hat{p}_D$, for $\varepsilon \to 0$, is an optimal defender policy.*

The value of any state $s \in \mathbf{S}$ is bounded above by β. From Lemma 6.1 we know that the sequence $V^{(k)}(s)$ is a monotonically increasing sequence. By the monotone convergence theorem (Royden & Fitzpatrick, 2010), a bounded and monotone sequence converges to the supremum, i.e., $\lim_{k\to\infty} V^{(k)}(s) \to V^\star(s)$, for all $s \in \mathbf{S}$. Thus the value iteration algorithm converges and the proof follows.

Theorem 6.4. *For any $\varepsilon > 0$, Algorithm 6.1 returns an ε-Nash equilibrium of the APT-DIFT game.*

Proof of Theorem 6.4 follows from Lemma 6.1, Theorem 6.4, and Definition 5.1. By Theorem 6.3, for any small value of ε, there exists a large enough K such that for $k > K$, $|v^{(k)}(s) - v^{(k+1)}(s)| < \varepsilon$ (Cauchy sequence). We note that, the approach only guarantees asymptotic convergence and it does not provide any guarantee on the rate of convergence. Let us consider an alternative approach by relaxing the problem after modifying step 2 of Algorithm 6.1 such that $v^{(k+1)}(s)$ is updated if

$$\max_{p_D\in\mathbf{P}_D} \min_{a\in\mathcal{A}_A(s)} \sum_{s'\in\mathbf{S}} \sum_{d\in\mathcal{A}_D(s)} p_D(s,d)P(s,a,d,s')V^{(k)}(s') > (1+\frac{\varepsilon}{\beta})v^{(k)}(s) \quad (21)$$

and remains unchanged otherwise. Then, we derive the following result on the termination time.

Lemma 6.5. *The update criteria in Eq. (21) converges to a value Vector V that satisfies $\max\{|V^{(k+1)}(s) - V^{(k)}(s)| : s \in \mathbf{S}\} \leqslant \varepsilon$ within $(N+4)\max_s\{\log(\frac{\beta}{v^0(s)})/\log(1+\frac{\varepsilon}{\beta})\}$ iterations, where $v^{(0)}(s)$ is the smallest positive value of $v^{(k)}(s)$ for $k = 0, 1, \ldots$, and N is the number of nodes in the IFG.*

Proof. From Eq. (21) and steps 2 and 5 of Algorithm 6.1, after T iterations of Algorithm 6.1, either Eq. (22) or Eq. (23) holds.

$$v^{(T)}(s) = v^{(T-1)}(s) \quad (22)$$

$$v^{(T)}(s) = \max_{p_D\in\mathbf{P}_D} \min_{a\in\mathcal{A}_A(s)} \sum_{s'\in\mathbf{S}} \sum_{d\in\mathcal{A}_D(s)} p_D(s,d)P(s,a,d,s')v^{(T-1)}(s')$$

$$> (1+\frac{\varepsilon}{\beta})v^{(T-1)}(s) \quad (23)$$

Thus $v^{(T)}(s) \geqslant (1+\frac{\varepsilon}{\beta})^T v^{(0)}(s)$. As a result, for each $s \in \mathbf{S}$, $v(s)$ will be updated at most $max_s\{\log(\frac{\beta}{v^{(0)}(s)})/\log(1+\frac{\varepsilon}{\beta})\}$ times. Additionally, in every iteration of Algorithm 6.1, value of at least one state is updated. This proves the upper bound on the number of iterations as we have $|\mathbf{S}| = N+4$. Also, after convergence, by Eq. (21), $|v^{(k+1)}(s) - v^{(k)}(s)| \leqslant \frac{\varepsilon}{\beta}v^{(k)}(s) \leqslant \varepsilon$. The second inequality holds as $v^{(k)}(s) \leqslant \beta$. □

Lemma 6.5 presents a trade-off between the desired accuracy of the value vector and the number of iterations.

Theorem 6.6 presents the computational complexity of Algorithm 6.1 for each iteration.

Theorem 6.6. *Consider the APT-DIFT game with N number of nodes in the IFG and q number of destination nodes. Let $\mathcal{A}_A$ and $\mathcal{A}_D$ be the action sets of APT and DIFT, respectively. Every iteration of Algorithm 6.1, i.e., steps 2–7, has computational complexity $O((N-q+1)^2|\mathcal{A}_A \parallel \mathcal{A}_D|)$.*

Proof. Every iteration of Algorithm 6.1 involves computation of the value vector. This involves solving, for every state $s \in \mathbf{S} \setminus \{\{\phi, \tau_A, \tau_B\} \cup \mathcal{D}\}$, a linear program of the form:

Maximize $V(s)$

Subject to : (1) $\sum_{d\in\mathcal{A}_D(s)} p_D(s,d) = 1$
(2) $p_D(s,d) \geqslant 0$, *for all* $d \in \mathcal{A}_D(s)$
(3) $V(s) \leqslant \sum_{d\in\mathcal{A}_D(s)} Q(s,d,a)p_D(s,d)$, *for all* $a \in \mathcal{A}_A(s)$.

We note that $|\mathbf{S} \setminus \{\{\phi, \tau_A, \tau_B\} \cup \mathcal{D}\}| = N-q+1$. The above linear program has complexity of $O((N-q+1)|\mathcal{A}_A \parallel \mathcal{A}_D|)$ (Boutilier, Dean, & Hanks, 1999). Thus solving for all $(N-q+1)$ states has a total complexity of $O((N-q+1)^2|\mathcal{A}_A \parallel \mathcal{A}_D|)$. □

From Lemma 6.5 and Theorem 6.6, the total computational complexity of Algorithm 6.1 to converge to an ε-Nash equilibrium is $O((N+4)\max_s\{\log(\frac{\beta}{v^0(s)})/\log(1+\frac{\varepsilon}{\beta})\}(N-q+1)|\mathcal{A}_A \parallel \mathcal{A}_D|)$ and this presents a selection criteria for the parameter ε.

We note that, Lemma 6.5 guarantees finite time convergence of Algorithm 6.1 with the modification suggested in Eq. (21). However, when the IFG is acyclic, the state space of the APT-DIFT game is acyclic (Theorem 6.8) and a finite time convergence of Algorithm 6.1 can be achieved even without any modification. Henceforth, the following assumption holds.

Assumption 6.7. The IFG $\mathcal{G}$ is acyclic.

The IFG obtained from the system log may not be acyclic in general. However, when the IFG is a cyclic graph, one can obtain an acyclic representation of the IFG using the node versioning technique proposed in Hossain et al. (2018). Throughout this subsection, we consider directed acyclic IFGs rendered using the approach in Hossain et al. (2018) and hence Assumption 6.7 is non-restrictive.

Theorem below presents a termination condition of the APT-DIFT game when the IFG is acyclic.

Theorem 6.8. *Consider the APT-DIFT game. Let Assumption 6.7 holds and N be the number of nodes in the IFG. Then the state space of the APT-DIFT game is acyclic and the game terminates in at most $N+4$ number of steps.*

Proof. Consider any arbitrary strategy pair (p_A, p_D). We first prove 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

Algorithm 6.2 Value iteration algorithm to find NE strategy for APT-DIFT game under Assumption 6.7

Input: APT-DIFT game with state space $\mathbf{S}$, destination set $\mathcal{D}$, action space $\mathcal{A}_D$, $\mathcal{A}_A$, payoff parameter β, transition probabilities $FN(\cdot)$, $FP(\cdot)$
Output: Value vector $V^\star$ and defender strategy $p_D^\star$

1: Find the topological ordering of the state space graph, $\mathcal{S}$
2: Using $\mathcal{S}$, obtain the set of nodes corresponding to hierarchical levels $L_1, L_2, \ldots, L_M$
3: Initialize value vector $V(s) \leftarrow 0$, for all $s \in \mathbf{S}$ and $V(s') \leftarrow \beta$ for $s' \in \{\tau_A, \phi\}$, $V(s'') \leftarrow 0$ for all $s'' \in \mathbf{S} \setminus \{\tau_A, \phi\}$
4: **for** $k \in \{M-1, M-2, \ldots, 1\}$ **do**
5: **for** $s \in L_k$ **do**
6: $V(s) \leftarrow \max_{p_D \in \mathbf{P}_D} \min_{a \in \mathcal{A}_A(s)} \sum_{s' \in \mathbf{S}} \sum_{d \in \mathcal{A}_D(s)} p_D(s,d) P(s,a,d,s') V(s')$
7: **end for**
8: $k \leftarrow k+1$
9: **end for**
10: **return** Value vector $V^\star \leftarrow V$
11: Compute DIFT strategy $p_D^\star$, $p_D^\star(s) \leftarrow \arg\max_{p_D \in \mathbf{P}_D} \min_{a \in \mathcal{A}_A(s)} \sum_{s' \in \mathbf{S}} \sum_{d \in \mathcal{A}_D(s)} p_D(s,d) P(s,a,d,s') V^\star(s')$

states v_0, ϕ, τ_A, and τ_B. We note that a state $s \in \{\phi, \tau_A, \tau_B\} \cup \mathcal{D}$ does not lie in a cycle in $\mathbf{S}$ as s is an absorbing state and hence have no outgoing edge, i.e., $\mathcal{A}_A(s) = \mathcal{A}_D(s) = \emptyset$. The state v_0 does not lie in a cycle in $\mathbf{S}$ as there are no incoming edges to v_0. This concludes that a state $s \in \{v_0, \phi, \tau_A, \tau_B\} \cup \mathcal{D}$ 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 some states in $v_{q+1}, \ldots, v_N$, since $\mathcal{D} = \{v_1, v_2, \ldots, v_q\}$. Recall that states $v_1, \ldots, v_N$ correspond to nodes 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 $|\mathbf{S}| = N+4$, $T \leqslant N+4$. □

From Lemma 6.5 and Theorem 6.6, the total computational Theorem 6.8, we propose a value iteration algorithm with guaranteed finite time convergence. We will use the following definition in our approach.

For a directed acyclic graph, topological ordering of the node set is defined below.

Definition 6.9 (*Kahn (1962)*)**.** 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 acyclic graph with vertex set B and edge set E there exists an algorithm of complexity $O(|B| + |E|)$ to find the topological order (Kahn, 1962). Using the topological ordering, one can find a hierarchical level partitioning of the nodes of a directed acyclic graph. Let $\mathcal{S}$ be the topologically ordered set of nodes of $\mathbf{S}$. Let the number of hierarchical levels associated with $\mathcal{S}$ be M, say $L_1, L_2, \ldots, L_M$. Then $L_1 = v_0$, $L_M = \{\phi, \tau_A, \tau_B\} \cup \mathcal{D}$. Moreover, a state s is said to be in hierarchical level L_j if there exists an edge into s from a state s' which is in some level $L_{j'}$, where $j' < j$, and there is no edge into s from a state s'' which is in level $L_{j''}$, where $j'' > j$.

Algorithm 6.2 presents the pseudocode to solve the APT-DIFT game using the topological ordering and the hierarchical levels, when the IFG is acyclic.

Corollary 6.10. *Consider the APT-DIFT game and let $\mathcal{A}_A$, $\mathcal{A}_D$ be the action sets of APT, DIFT, respectively. Let the IFG is acyclic with N number of nodes and q number of destination nodes. Then, Algorithm 6.2 returns the value vector $V^\star$. Moreover, Algorithm 6.2 has computational complexity $O((N-q+1)^2 |\mathcal{A}_A \parallel \mathcal{A}_D|)$.*

Proof. Recall that under Assumption 6.7, the state space $\mathbf{S}$ of the APT-DIFT game is acyclic. Thus the topological ordering $\mathcal{S}$ and the hierarchical levels, $L_1, \ldots, L_M$, of $\mathbf{S}$ can be obtained in polynomial time (Kahn, 1962). We note that, values at the absorbing states, i.e., states at hierarchical level M, are $V(s) = \beta$ for $s \in \{\tau_A, \phi\}$ and $V(s') = 0$ for all $s' \in \{\tau_B\} \cup \mathcal{D}$. Using the hierarchical levels, the value vector can be computed in a dynamic programming manner starting from states in the last hierarchical level. Initially, using the values of the states at level M, the values of states at level $M-1$ can be obtained. Similarly, using values of states at level $M-1$ and level M, the values of states at level $M-2$ can be obtained and so on. By recursive computations we obtain the value vector $V^\star$ and the corresponding DIFT strategy $p_D^\star$.

Finding the topological ordering and the corresponding hierarchical levels of the state space graph has $O(|\mathbf{S}|^2)$ computations. The linear program associated with each state in the value iteration involves $O((N-q+1)|\mathcal{A}_A \parallel \mathcal{A}_D|)$ computations (See proof of Theorem 6.6). Since we need to compute values corresponding to $(N-q+1)$ states, complexity of Algorithm 6.2 is $O((N-q+1)^2 |\mathcal{A}_A \parallel \mathcal{A}_D|)$. □

6.2. Hierarchical Supervised Learning (HSL) algorithm

This subsection presents our approach to solve the APT-DIFT game when the game model is not fully known. It is often unrealistic to know the precise values of the false-positive rates and the false-negative rates of DIFT as these values are estimated empirically. More specifically, the transition probabilities of the APT-DIFT game are unknown. The traditional reinforcement learning algorithms (Q-learning) for constant-sum games with unknown transition probabilities (Hu & Wellman, 1998) assume perfect information, i.e., the players can observe the past actions and rewards of the opponents (Hu & Wellman, 1998).

On the other hand, dynamic programming-based approaches, including value iteration and policy iteration algorithms, require the transition probabilities in the computations. When $FN(\cdot)$ and $FP(\cdot)$ values are unknown, the optimization problem associated with the states in step 5 of Algorithm 6.1 and step 6 of Algorithm 6.2 cannot be solved. That is, in the LP

Problem 6.11. *Maximize $V(s)$*

$$\text{Subject to}: \quad (1) \quad \sum_{d \in \mathcal{A}_D(s)} p_D(s,d) = 1$$
$$(2) \quad p_D(s,d) \geqslant 0, \ \text{for all } d \in \mathcal{A}_D(s)$$
$$(3) \quad V(s) \leqslant \sum_{d \in \mathcal{A}_D(s)} Q(s,d,a) p_D(s,d), \ \text{for all } a \in \mathcal{A}_A(s),$$

the $Q(s,a,d)$ values are unknown, where

$$Q(s,a,d) = \sum_{s' \in \mathbf{S}} P(s,a,d,s') V(s'),$$

and $P(s,a,d,s')$ is defined in Eq. (1). To the best of our knowledge, there are no known algorithms, with guaranteed equilibrium convergence, to solve imperfect and incomplete stochastic game models as that of the APT-DIFT game presented in this paper. To this end, we propose a supervised learning-based approach to solve the APT-DIFT game. Our approach consists of two key steps.

(1) Training a neural network to approximate the value vector of the game for a given strategy pair, and
(2) A policy iteration algorithm to compute an ε-optimal NE strategy.

Our approach to solve the APT-DIFT game, when the rates of false-positives and false-negatives are unknown, utilizes the topological ordering and the hierarchical levels of the state space graph. We propose a hierarchical supervised learning (HSL)-based approach, that predicts the Q-values of the APT-DIFT game and then solve the LP for all the states (Problem 6.11) in a hierarchical manner, to approximate an NE strategy of the APT-DIFT game. In HSL, we utilize a neural network to obtain a mapping between the strategies of the players and the value vector. This mapping is an approximation, using which HSL presents an approach to compute an approximate equilibrium and evaluates the performance empirically using real attack datasets.

In order to predict the Q-values of the game, we train a neural network to learn the value vector of the game and use the trained model to predict the Q-values. The reasons to train an NN to learn the value vector instead of directly learning the Q-values are: (a) while the value vector is of dimension $|\mathbf{S}|$ the Q-values have dimension $|\mathbf{S}||\mathcal{A}_A||\mathcal{A}_D|$ and (b) generation of data samples for value vector is easy. The approach for data generation and training is elaborated below.

We note that, it is possible to simulate the APT-DIFT game and observe the final game outcome, i.e., the payoffs of the players. For a given (p_A, p_D), the value at state s, $V(s)$, is the payoff of the defender if the game originate at state s, i.e., $V(s) = U_D(s, p_A, p_D)$. Training the neural network for predicting the value vector of the APT-DIFT game consists of two steps.

(i) Generate random samples of strategy pairs, (p_A, p_D)
(ii) Simulate the APT-DIFT game for each of the randomly generated sample of strategy pair and obtain the values corresponding to all states.

The neural network takes as input the strategy pairs and outputs the value vector. The training is done using a multi-input, multi-output neural network, represented as $\mathcal{F} : X \to Y$, where $X \subseteq [0,1]^{|\mathcal{A}_A|} \times [0,1]^{|\mathcal{A}_D|}$ and $Y \subseteq \mathbb{R}^{|\mathbf{S}|}$. The neural network may not compute the exact value vector, however, it can approximate the value vector to arbitrary accuracy. Given a function $f(x)$ and $\xi > 0$, the guarantee is that by using enough hidden neurons it is always possible to find a neural network whose output $g(x)$ satisfies $|f(x) - g(x)| < \xi$, for all inputs x (Csáji et al., 2001). In other words, the approximation will be good to within the desired accuracy for every possible input. However, the training method does not guarantee that the neural network obtained at the end of the training process is one that satisfies the specified level of accuracy. Using the trained neural network we predict the Q-values of the game.

Lemma 6.12. *Consider the APT-DIFT game with state space* $\mathbf{S}$*. Let a neural network be trained using samples of strategy pairs* (p_A, p_D) *to predict the value vector* V *of the game such that the mean absolute error is within the desired tolerance of* $\zeta \leqslant 0.01$*. Then, the trained neural network also yield the* Q*-values,* $Q(s, a, d)$*, for all* $s \in \mathbf{S}$*,* $a \in \mathcal{A}_A$*, and* $d \in \mathcal{A}_D$*.*

Proof. Consider a neural network that is trained using enough samples of strategy pairs to predict the value vector of the game. Thus given a strategy pair (p_A, p_D), the neural network predicts $V = \{V(s) : s \in \mathbf{S}\}$. Consider an arbitrary state s. The Q-value corresponding to state s and action pair a, d for the APT and DIFT, respectively, is given by $Q(s, a, d) = \sum_{s' \in \mathbf{S}} P(s, a, d, s')V(s')$. For a strategy (p_A, p_D) with $p_A(s)$ such that $p_A(s, a) = 1$ and $p_D(s)$ such that $p_D(s, d) = 1$ gives

$$\begin{aligned} V(s) &= \sum_{s' \in \mathbf{S}} \sum_{a \in \mathcal{A}_A} \sum_{d \in \mathcal{A}_D} p_A(s, a) p_D(s, d) P(s, a, d, s')V(s') \\ &= \sum_{s' \in \mathbf{S}} P(s, a, d, s')V(s') \\ &= Q(s, a, d). \end{aligned}$$

Hence the Q-values of the game can be obtained using the neural network that is trained to predict the value vector by inputting a strategy pair (p_A, p_D) with $p_A(s, a) = 1$ and $p_D(s, d) = 1$, for any $s \in \mathbf{S}$, $a \in \mathcal{A}_A$, and $d \in \mathcal{A}_D$. □

Using the trained neural network we run a policy iteration algorithm. In the policy iteration, we update the strategies of both players, APT and DIFT, by solving the stage games in a dynamic programming manner. The details of the algorithm are presented below.

Algorithm 6.3 HSL Algorithm for APT-DIFT game

Input: APT-DIFT game with state space $\mathbf{S}$, destination set $\mathcal{D}$, action sets $\mathcal{A}_A$, $\mathcal{A}_D$, payoff parameter β
Output: Value vector $\hat{V}$ and defender strategy $\hat{p}_D$

1: Generate random samples of (p_A, p_D) and value vector
2: Train $\mathcal{F}$ using the data set from Step 1
3: Find the topological ordering of the state space graph, $\mathcal{S}$
4: Obtain the set of nodes corresponding to hierarchical levels $L_1, L_2, \ldots, L_M$
5: Initialize $V(\tau_A) \leftarrow \beta$, $V(\phi) \leftarrow \beta$, $V(\tau_B) \leftarrow 0$, and $V(s) \leftarrow 0$ for $s \in \mathcal{D}$
6: Initialize randomly $(\hat{p}_A^{(M-1)}, \hat{p}_D^{(M-1)})$
7: **for** $k \in \{M-1, M-2, \ldots, 1\}$ **do**
8: **for** $s \in L_k$ **do**
9: **for** $a \in \mathcal{A}_A(s)$, $d \in \mathcal{A}_D(s)$ **do**
10: Update $(\hat{p}_A^{(k)}, \hat{p}_D^{(k)})$ such that $\hat{p}_A^{(k)}(s, a) = 1$ and $\hat{p}_D^{(k)}(s, d) = 1$
11: Predict the value vector $\hat{V}$ using the neural network $\mathcal{F}$, $\hat{V} \leftarrow \mathcal{F}(\hat{p}_A^{(k)}, \hat{p}_D^{(k)})$
12: Assign $\hat{Q}(s, a, d) \leftarrow \hat{V}(s)$
13: **end for**
14: Assign $Q(s, a, d) \leftarrow \hat{Q}(s, a, d)$ and solve Problem 6.11 for s to obtain $V(s)$ and $p_D(s)$
15: Update $\hat{p}_D^{(k)}$ such that $\hat{p}_D^{(k)}(s) \leftarrow p_D(s)$
16: Find the action corresponding to the minimum entry of the vector $\hat{Q}(s, a, d)\,\hat{p}_D^{(k)}(s, d)$, say $\hat{a}$
17: Update $\hat{p}_A^{(k)}$ such that $\hat{p}_A^{(k)}(s, \hat{a}) \leftarrow 1$
18: **end for**
19: **end for**
20: **return** $\hat{p}_D \leftarrow \hat{p}_D^{(k)}$

Algorithm 6.3 presents the pseudocode for our HSL algorithm to solve the APT-DIFT game. Initially we generate random samples of strategy pairs (p_A, p_D) and value vector V and train a neural network. Using the trained neural network, we propose a strategy iteration algorithm. As Assumption 6.7 holds, the state space graph $\mathbf{S}$ is acyclic. Thus one can compute the topological ordering (Step 3 and the hierarchical levels (Step 4 of $\mathbf{S}$ in polynomial time. The values at the absorbing states are known and are hence set as $V(\tau_A) = V(\phi) = \beta$, $V(\tau_B) = 0$, and $V(s) = 0$ for all $s \in \mathcal{D}$. As the values of the states at level M are known, the algorithm begins with level $M-1$. Initially the strategies of APT and DIFT are randomly set to $\hat{p}_A^{(M-1)}$ and $\hat{p}_D^{(M-1)}$, respectively. Then we compute the Q-values of all the states using the trained neural network, following the hierarchical order. Due to the hierarchical structure of the state space, computation of Q-value of a state in jth hierarchical level depends only on the states that are at levels higher than j. Consider level $M-1$ and an arbitrary state s in level $M-1$. The Q-values of s are predicted using the trained neural network by selecting suitable deterministic strategy pairs. That is, for predicting $Q(s, a, d)$, choose strategies such that $\hat{p}_A^{(M-1)}(s, a) = 1$ and $\hat{p}_D^{(M-1)}(s, d) = 1$ as input to the neural network. Using the Q-values of state s, solve the LP and obtain value vector, DIFT

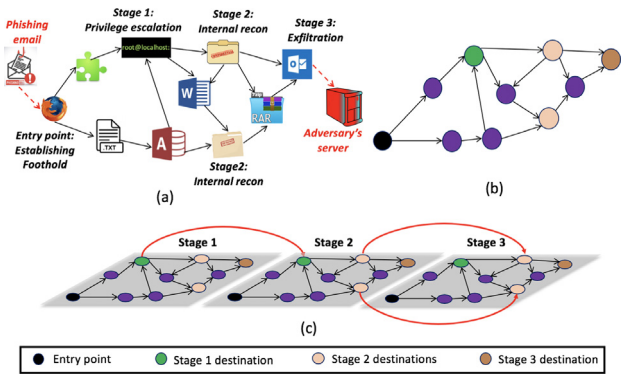

Fig. 1. An example of a multi-stage APT attack scenario (Fig. 1(a)). An illustrative diagram of the IFG $\mathcal{G}$ (Fig. 1(b)) and the corresponding multi-stage IFG $\mathcal{G}_m$ (Fig. 1(b)) that consists of three stages of the attack, i.e., $m = 3$. The propagation of the attack from stage j of the attack to stage $(j+1)$ is captured in $\mathcal{G}_m$ by connecting the destination nodes $\mathcal{D}_j$ in stage j to their respective nodes in stage $(j+1)$, for $j = 1, 2$.

strategy, and the optimal action of the APT. The strategies $\hat{p}_A^{(M-1)}$ and $\hat{p}_D^{(M-1)}$ are updated using the output of the LP such that the $\hat{p}_A^{(M-1)}(s)$ and $\hat{p}_D^{(M-1)}(s)$ corresponds to an NE strategy. Once the strategies of both players are updated for all the states in level $M-1$, the process continues for level $M-2$ and so on.

In HSL algorithm, the prediction of Q-values using the neural network (Step 8–12) corresponding to states in a particular level can be run parallel. To elaborate, let there are x number of states in level L_j. Then select an action pair (a, d) corresponding to each state and run the prediction step of the algorithm. Thus, every run of the neural network can predict x number of Q-values, for x different states.

Remark: APT attacks typically consist of multiple stages, e.g., initial compromise, internal reconnaissance, foothold establishment, and data exfiltration, with each stage having a specific set of targets. To capture the multi-stage nature of the attack, we construct a multi-stage IFG, $\mathcal{G}_m$, from the IFG $\mathcal{G}$. Consider an attack that consists of m attack stages with destinations of stage j denoted as $\mathcal{D}_j$. Then we duplicate m copies of the IFG $\mathcal{G}$ such that nodes in $\mathcal{D}_j$ in the jth copy of $\mathcal{G}$ is connected to respective nodes in $\mathcal{D}_j$ in the $(j+1)$th copy, for $j \in \{1, \ldots, m\}$. Also, set $\mathcal{D}_m = \mathcal{D}$. The construction of $\mathcal{G}_m$ guarantees that any path that originate in set λ in the first copy and terminate in $\mathcal{D}$ in the mth copy is a feasible attack path. Fig. 1 shows schematic diagram of a multi-stage IFG. For notational brevity the paper presents the single-stage attack case, i.e., $m = 1$. All the algorithm and results presented in this paper also apply to the case of multi-stage attack using $\mathcal{G}_m$ as the IFG.

7. Experiments and discussions

In this section we test and validate Algorithms 6.1, 6.2 and HSL algorithm (Algorithm 6.3) on two real world attack datasets. First we provide the details on the attack datasets and explain the construction of IFGs corresponding to each attack from their respective system log data. Then we discuss our experiments and present the results.

7.1. Attack datasets

We use two attack datasets in our experiments. The first dataset is built from a nation state attack (Brenner, 2009) and the second is related to a ransomware attack (Mohurle & Patil, 2017). Each attack was executed individually in a computer running *Linux* operating system and system logs were recorded through RAIN system (Ji et al., 2017). System logs contain records related to both malicious and benign information flows.

7.1.1. Nation state attack

Nation state attack (a state-of-the-art APT attack) is a three day adversarial engagement orchestrated by US DARPA red-team during the evaluation of RAIN system. Attack campaign was designed to steal sensitive proprietary and personal information from the victim system. In our experiments we used the system logs recorded during the day 1 of the attack. The attack consists of four key stages: initial compromise, internal reconnaissance, foothold establishment, and data exfiltration. Our experiments considered the first stage of the attack, i.e., initial compromise stage, where APT used spear-phishing to lead the victim user to a website that was hosting ads from a malicious web server. After navigating to the website, APT exploited a vulnerability in the Firefox browser and compromised the Firefox.

7.1.2. Ransomware attack

Ransomware attack campaign was designed to block access to the files in *./home* directory and demand a payment from the victim in exchange for regranting the access. The attack consists of three stages: privilege escalation, lateral movement of the attack, and encrypting and deleting *./home* directory. Upon reaching the final stage of the attack, APT first opened and read all the files in the *./home* directory of the victim computer. Then APT wrote the content of the files in *./home* into an encrypted file named *ransomware.encrypted*. Finally, APT deleted the *./home* directory.

7.2. Construction of IFG

Direct conversion of the system logs into IFG typically results in coarse graphs with large number of nodes and edges as it includes all the information flows recorded during the system execution. Majority of these information flows are related to the system's background processes (noise) which are unrelated to the actual attack campaign and it is computationally intensive to run the proposed algorithms on a state space induced by such a coarse graph. Hence, we use the following pruning steps to prune the coarse graph without losing any attack related causal information flow dependencies.

1. When multiple edges with same directed orientation exist between two nodes in the coarse IFG, combine them to a single directed edge. For example, consider a scenario where multiple "read" system calls are recorded between a file and a process. This results in multiple edges between the two nodes of the resulting coarse IFG. Our APT-DIFT game formulation only requires to realize the feasibility of transferring information flows between pairs of processes and files. Hence, in scenarios similar to the above example, we collapse all the multiple edges between the two nodes in the coarse IFG into a single edge.
2. Find all the nodes in coarse IFG that have at least one information flow path from an entry point of the attack to a target of the attack. When attack consists of multiple stages find all the nodes in coarse IFG that have at least one information flow path from a destination of stage j to a destination of a stage $j+1$, for all $j \in \{1, \ldots, m-1\}$.
3. From coarse graph, extract the subgraph corresponding to the entry points, destinations, and the set of nodes found in Step 2.

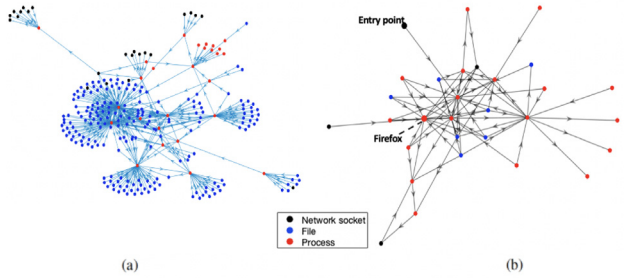

Fig. 2. Relevant attack information of nation state attack: (a) coarse IFG and (b) pruned IFG. Nodes of the graph are color coded to illustrate their respective types (network socket, file, and process). A network socket is identified as the entry points of the nation state attack. Target of the attack is *Firefox* process. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4. Group all the nodes that correspond to files of a single directory to a single node related to the parent file directory. For example, assume the resulting coarse IFG has three files, *./home/inventory/prices.xlsx*, *./home/vendors/contacts/addresses.doc*, and *./home/vendors/ledger.db*, that are uniquely identified by their respective file paths. In this case we will group all three files, *prices.xlsx*, *addresses.doc*, and *ledger.db* under one super-node corresponding to the parent directory *./home*. Incoming and outgoing edges associated with each of the three files are then connected to the new node *./home*. If any subset of the new edges connected to *./home* induce multiple edges with same orientation to another node in the IFG, then follow step (1). It is also possible to group the files into respective subdirectories such as *./home/inventory/* and *./home/vendors/* given in the example. Such groupings will facilitate much finer-grained security analysis at the cost of larger IFGs which require more computation resources to run the proposed APT-DIFT game algorithms presented in this paper.
5. If the resulting graph after Steps 1−4 contains any cycles use *node versioning* techniques (Hossain et al., 2018) to remove cycles while preserving the information flow dependencies in the graph.

Steps 2 and 3 are done using upstream, downstream, and point-to-point stream pruning techniques mentioned in Ji et al. (2017). The resulting information flow graph is called pruned IFG. We tested the proposed algorithms on the state spaces of APT-DIFT games corresponding to these pruned IFGs.

For the nation state attack, initial conversion of the system logs into an IFG resulted in a coarse graph with 299 nodes and 404 edges which is presented in Fig. 2(a). We used steps 1 to 4 explained in Section 7.2 to obtain a pruned IFG with 30 nodes and 74 edges. A network socket connected to an untrusted IP address was identified as an entry point of the attack and the target of the attack is Firefox process. Fig. 2(b) shows the pruned IFG of nation state attack. In the IFG, there are 21 information flow paths from the entry point to the target Firefox.

For the ransomware attack, direct conversion of the system logs resulted in a coarse IFG with 173 nodes and 482 edges which is presented in Fig. 3(a). We pruned the resulting coarse IFG using the steps given in Section 7.2. The pruned IFG of ransomware attack consists of 18 nodes and 29 edges. Two network sockets that indicate series of communications with external IP addresses in the recorded system logs were identified as the entry points of the attack. Fig. 3(b) illustrates the pruned IFG of ransomware attack with annotated targets */usr/bin/sudo*, */bin/bash*, */home*.

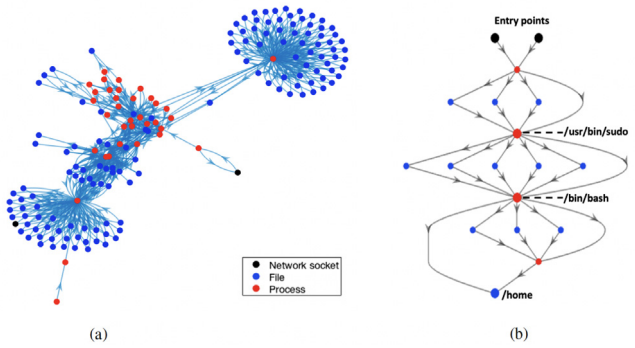

Fig. 3. Relevant attack information of ransomware attack: (a) coarse IFG and (b) pruned IFG. Nodes of the graph are color coded to illustrate their respective types (network socket, file, and process). Two network sockets are identified as the entry points of the ransomware attack. Targets of the attack (*/usr/bin/sudo*, */bin/bash*, */home*) are labeled in the graph. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7.3. Experiments and results

We evaluated the proposed algorithms by solving the APT-DIFT game on the pruned IFGs of the nation state and ransomware attacks, shown in Figs. 2 and 3, respectively. We note that, the IFG of the nation state attack is cyclic and that of the ransomware attack is acyclic. We implemented Algorithm 6.1 on the pruned IFG of the nation state attack. Fig. 4(a) shows the convergence of the value $V^{(k)}(s_0)$ in APT-DIFT game with iteration number k. The threshold value of the maximum absolute error, i.e., ε, in Algorithm 6.1 was set to 10^{-7}. Fig. 4(b) shows that maximum absolute error, $\max_{s\in\mathbf{S}} |V^{(k)}(s) - V^{(k-1)}(s)|$ where $k = 1, 2, \ldots$, monotonically decreases with k. At iteration $k = 32$, $\max_{s\in\mathbf{S}} |V^{(k)}(s) - V^{(k-1)}(s)| = 7.79 \times 10^{-8}$.

We implemented HSL algorithm (Algorithm 6.3) on the pruned IFG of the ransomware attack and results are given in Figs. 5 and 6. We used a sequential neural network with two dense layers to learn the value vector for a given strategy pair of APT and DIFT. Each dense layer consists of 1000 neurons and ReLU activation function. The dataset consist tuples of APT and DIFT strategy pair and corresponding value vector of APT-DIFT game. In our experiments, we generated data samples that consist of randomly generated deterministic APT policies. We randomly generated DIFT's policies such that 40% strategies are stochastic (mixed) and 60% are deterministic. For each randomly generated APT and DIFT strategy pair, the corresponding value vector was computed. We partitioned the generated data into three disjoint categories (i) training data, (ii) validation data, and (iii) testing data. The training data is used to train the model (in our case a Neural Network), the validation data is used to fine tune the parameters of the model, and the testing data is used to evaluate whether the learned parameter values using the first two partitions are 'correct' (Mitchell, 1997). We used stochastic gradient descent optimizer to train the neural network. In each experiment trial the neural network was trained for 100 episodes and validation error was set to $< 1\%$.

Fig. 5 compares the value vector, i.e., $\{V(s_i)$: for all $s_i \in \mathbf{S}\}$, computed using HSL algorithm (Algorithm 6.3) when the transition probabilities are unknown with two instances with known transition probabilities: (i) the exact/optimal value vector and (ii) value vector computed under a scenario in which both players follow a greedy strategy. We know that when the transition probabilities are known and the IFG is acyclic, Algorithm 6.2 returns the optimal/exact value vector of the game, i.e., $V^\star$. We

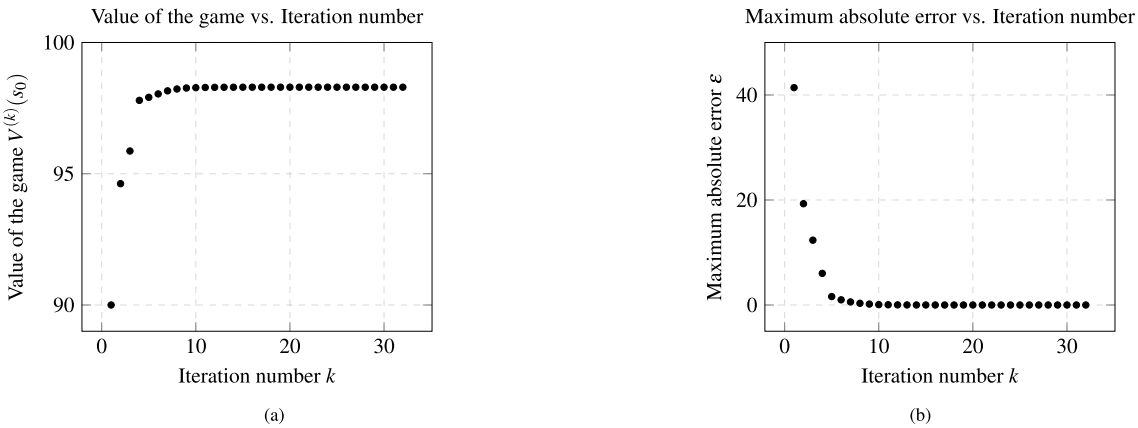

Fig. 4. Fig. 4(a) plots the value of APT-DIFT game $V^{(k)}(s_0)$ corresponding to pruned IFG of nation state attack given in Fig. 2, computed at iterations $k = 1, 2, \ldots, 32$ in Algorithm 6.1. Fig. 4(b) plots the maximum absolute error $\varepsilon = \max_{s\in\mathbf{S}} |V^{(k)}(s) - V^{(k-1)}(s)|$, for $k = 1, 2, \ldots, 32$. The payoff parameter is set to $\beta = 100$.

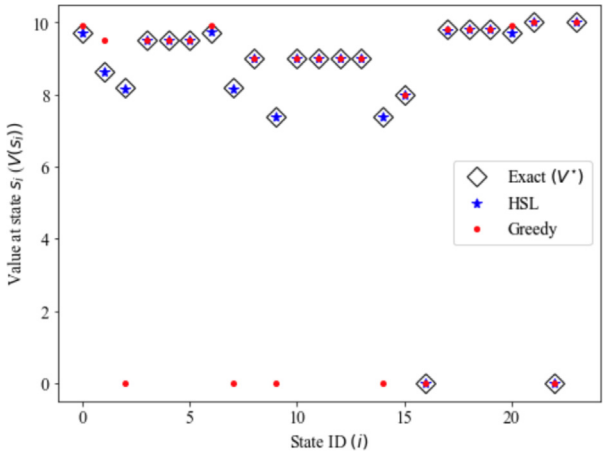

Fig. 5. Comparison of value at each state, $V(s_i)$ for all $s_i \in \mathbf{S}$, obtained using HSL algorithm (Algorithm 6.3) and greedy algorithm (Greedy) with exact/optimal value vector for the APT-DIFT game of ransomware attack data. Here, $\beta = 10$. In the greedy algorithm, at each state s_i, DIFT selects an action $\bar{d}$ that gives maximum $Q(s_i, a, \bar{d})$ for any $a \in \mathcal{A}_A(s_i)$ and APT chooses an action $\bar{a}$ that gives maximum $Q(s_i, \bar{a}, d)$ for any $d \in \mathcal{A}_D(s_i)$. State ID's $i = 20, 21, 22$, and 23 represent the states s_0, ϕ, τ_A, and τ_B, respectively. Values corresponding to HSL algorithm (Algorithm 6.3) are averaged over 100 independent trials.

computed the exact/optimal value at each state, i.e., $V^\star(s_i)$'s, using Algorithm 6.2. We also implemented a greedy algorithm (Greedy) in which at each state s_i, DIFT selects an action $\bar{d}$ that attains the maximum $Q(s_i, a, \bar{d})$ for any $a \in \mathcal{A}_A(s_i)$ and APT selects an action $\bar{a}$ that yields the maximum $Q(s_i, \bar{a}, d)$ for any $d \in \mathcal{A}_D(s_i)$. Values corresponding to HSL algorithm (Algorithm 6.3) are averaged over 100 independent trials. Average standard deviation for the 100 trials of HSL is 0.017. The mean absolute error, i.e., $\sum_{s_i\in\mathbf{S}} |V^\star(s_i) - V(s_i)|/|\mathbf{S}|$, for the HSL and greedy algorithms are 0.00680 and 1.35575, respectively. The state space of the APT-DIFT game for the ransomware attack has 23 states, out of which 5 are absorbing states (ϕ, τ_A, τ_B and two destinations $\mathcal{D}$). Since the values at the absorbing states are known, we need to estimate values at 18 states. As shown in Fig. 5 the value vector obtained using greedy does not converge for 8 states out of the 18 non-absorbing states, i.e., it does not converge for 44.4% of the states. The solution of the HSL, on the other hand, converges for all the 18 states. Thus our empirical results show that the approximate value vector computed using HSL coincides with the optimal value vector, i.e., NE. The value vector computed using the greedy algorithm, on the other hand, does not converge to the optimal value for many states and the mean absolute error for greedy is ≈200 times of the HSL algorithm.

In Fig. 6(a) we analyze the sensitivity of estimated value vector to the variations of payoff parameter, β by plotting the mean absolute error μ between actual value vector V and estimated value vector $\hat{V}$, i.e., $\mu = \sum_{s\in\mathbf{S}} |V(s) - \hat{V}(s)|/|\mathbf{S}|$, with respect to β. In this experiment 10^5 training data samples were used in the HSL algorithm. The results show that $\hat{V}$ values are close and consistent with V values when β parameter takes smaller values and variations between $\hat{V}$ and V is increased when β takes larger values. A reason for this behavior can be the numerical unstability associated with the training and estimating tasks done in the neural network model used in HSL algorithm. In order to study the effect of the length of training data samples on the estimated value vector we plot μ against number of training data samples used in HSL algorithm in Fig. 6(b). $\beta = 50$ was used in the experiments. The results show that μ decreases with the number of training data samples used in HSL algorithm as the increased number of training data samples improves the learning in neural network model.

8. Conclusion

This paper studied detection of Advanced Persistent Threats (APTs) using a Dynamic Information Flow Tracking (DIFT)-based detection mechanism. We modeled the strategic interaction between the APT and DIFT as a non-cooperative stochastic game with total reward structure. The APT-DIFT game has imperfect information as both APT and DIFT are unaware of the actions of the opponent. Also, our game model incorporates the false-positives and false-negatives generated by DIFT. We considered two scenarios of the game (i) when the transition probabilities, i.e., rates of false-positives and false-negatives, are known to both players and (ii) when the transition probabilities are unknown to both players. For case (i), we proposed a value iteration-based algorithm and proved convergence and complexity. For case (ii), we proposed Hierarchical Supervised Learning (HSL), a supervised learning-based algorithm that utilizes the hierarchical structure of the state space of the APT-DIFT game, that integrates a neural network with a policy iteration algorithm to compute an approximate equilibrium of the game. We validated our results using real attack datasets for nation state attack and ransomware attack collected using the RAIN framework. We compared the performance of the HSL algorithm when the transition probabilities are unknown with instances with known transition probabilities and demonstrated that HSL algorithm converges to a solution close to optimal (i.e., optimal value vector) while the value vector

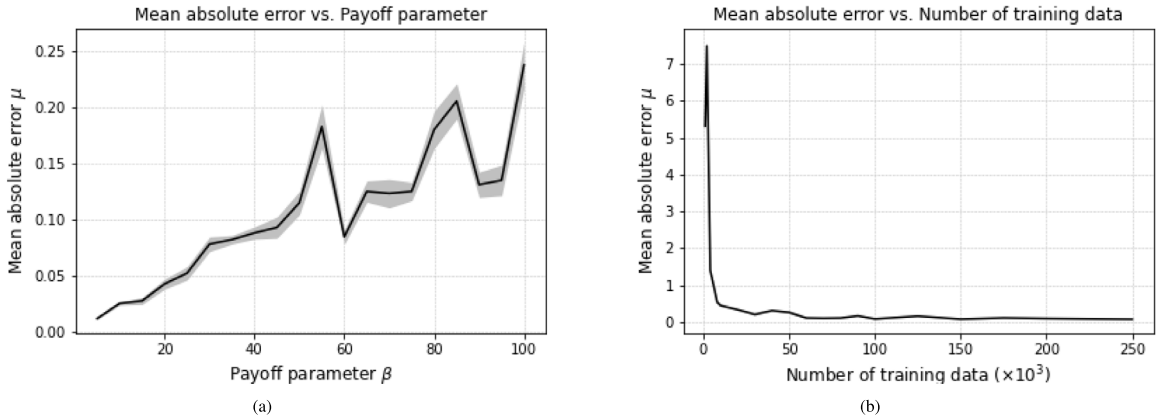

Fig. 6. Fig. 6(a) shows the mean absolute error μ between actual value vector V and estimated value vector $\hat{V}$ (using neural network) for different payoff parameter values $\beta = 5, 10, \ldots, 100$. We used 10^5 training data samples in the HSL algorithm (Algorithm 6.3). Fig. 6(b) shows variation in μ with respect to the number of training data samples used in HSL algorithm. In this experiment $\beta = 50$. Each data point in both cases are calculated by averaging results over 10 independent trials. Bars in each data point shows the standard errors associated with the μ values obtained in the different trials. Pruned IFG corresponding to the ransomware attack in Fig. 3 was used in both cases.

obtained using greedy does not converge to optimal for 44.4% of the states and the mean absolute error is almost 200 times that of the HSL.

Acknowledgment

Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF or its federal agency and industry partners.

Appendix

Proof of Proposition 5.2. It is known that the class of zero-sum, finite, stochastic games with nonzero stopping (termination) probability has Nash equilibrium in stationary strategies (Shapley, 1953). Note that, the APT-DIFT game is a stochastic game with finite state space and finite action spaces for both players. Moreover, the transition probabilities in the game $FP(\cdot)$ and $FN(\cdot)$ are such that $0 < FP(\cdot) < 1$ and $0 < FN(\cdot) < 1$. Thus the stopping probabilities of the APT-DIFT game are nonzero, for all strategy pairs (p_A, p_D) except for a deterministic policy in which the defender does not perform security analysis at any state. However, such a policy is trivially irrelevant to the game as the defender is idle and essentially not participating in the game. As the result for zero-sum games also hold for constant-sum games, from Shapley (1953) it follows that there exists an NE for the APT-DIFT game. □

References

Ahmadi, M., Cubuktepe, M., Jansen, N., Junges, S., Katoen, J.-P., & Topcu, U. (2018). The partially observable games we play for cyber deception. arXiv:1810.00092.

Ahmadi, M., Viswanathan, A. A., Ingham, M. D., Tan, K., & Ames, A. D. (2020). Partially observable games for secure autonomy. In *2020 IEEE security and privacy workshops (SPW)* (pp. 185–188).

Amir, R. (2003). Stochastic games in economics and related fields: An overview. *Stochastic Games and Applications*, 455–470.

Bencsáth, B., Pék, G., Buttyán, L., & Felegyhazi, M. (2012). The cousins of stuxnet: Duqu, flame, and Gauss. *Future Internet*, *4*(4), 971–1003.

Boutilier, C., Dean, T., & Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, *11*, 1–94.

Brenner, S. W. (2009). *Cyberthreats: the emerging fault lines of the nation state*. Oxford University Press.

Clause, J., Li, W., & Orso, A. (2007). Dytan: a generic dynamic taint analysis framework. In *International symposium on software testing and analysis* (pp. 196–206).

Csáji, B. C., et al. (2001). Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, *24*(48), 7.

Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.-G., Cox, L. P., et al. (2014). TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems*, *32*(2), 5.

Hossain, M. N., Wang, J., Sekar, R., & Stoller, S. D. (2018). Dependence-preserving data compaction for scalable forensic analysis. In *USENIX security symposium* (pp. 1723–1740).

Hu, J., & Wellman, M. P. (1998). Multiagent reinforcement learning: Theoretical framework and an algorithm.. *International Conference on Machine Learning*, *98*, 242–250.

Hu, J., & Wellman, M. P. (2003). Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, *4*, 1039–1069.

Huang, L., & Zhu, Q. (2019). Adaptive strategic cyber defense for advanced persistent threats in critical infrastructure networks. *ACM SIGMETRICS Performance Evaluation Review*, *46*(2), 52–56.

Huang, L., & Zhu, Q. (2020). A dynamic games approach to proactive defense strategies against advanced persistent threats in cyber-physical systems. *Computers & Security*, *89*, Article 101660.

Jang-Jaccard, J., & Nepal, S. (2014). A survey of emerging threats in cybersecurity. *Journal of Computer and System Sciences*, *80*(5), 973–993.

Ji, Y., Lee, S., Downing, E., Wang, W., Fazzini, M., Kim, T., et al. (2017). RAIN: Refinable attack investigation with on-demand inter-process information flow tracking. In *ACM SIGSAC conference on computer and communications security* (pp. 377–390).

Kahn, A. B. (1962). Topological sorting of large networks. *Communications of the ACM*, *5*(11), 558–562.

Lajevardi, A. M., & Amini, M. (2019). A semantic-based correlation approach for detecting hybrid and low-level APTs. *Future Generation Computer Systems*, *96*, 64–88.

Littman, M. L. (2001). Value-function reinforcement learning in Markov games. *Cognitive Systems Research*, *2*(1), 55–66.

Lye, K.-w., & Wing, J. M. (2005). Game strategies in network security. *International Journal of Information Security*, *4*(1–2), 71–86.

Misra, S., Moothedath, S., Hosseini, H., Allen, J., Bushnell, L., Lee, W., et al. (2019). Learning equilibria in stochastic information flow tracking games with partial knowledge. *IEEE Conference on Decision and Control*, 4053–4060.

Mitchell, T. (1997). Machine learning.

Mohurle, S., & Patil, M. (2017). A brief study of wannacry threat: Ransomware attack 2017. *International Journal of Advanced Research in Computer Science*, *8*(5).

Moothedath, S., Sahabandu, D., Allen, J., Clark, A., Bushnell, L., Lee, W., et al. (2020a). A game-theoretic approach for dynamic information flow tracking to detect multi-stage advanced persistent threats. *IEEE Transactions on Automatic Control*, *65*(12), 5248–5263.

Moothedath, S., Sahabandu, D., Allen, J., Clark, A., Bushnell, L., Lee, W., et al. (2020b). Dynamic information flow tracking for detection of advanced persistent threats: A stochastic game approach. *IEEE Transactions on Automatic Control* (conditionally accepted).

Najim, K., Poznyak, A. S., & Gomez, E. (2001). Adaptive policy for two finite Markov chains zero-sum stochastic game with unknown transition matrices and average payoffs. *Automatica*, *37*(7), 1007–1018.
Prasad, H. L., Prashanth, L. A., & Bhatnagar, S. (2015). Two-timescale algorithms for learning Nash equilibria in general-sum stochastic games. In *International conference on autonomous agents and multiagent systems* (pp. 1371–1379).
Rass, S., König, S., & Schauer, S. (2020). Defending against advanced persistent threats using game-theory. *PLoS One, Public Library of Science*, *12*(1), 1–43.
Royden, H., & Fitzpatrick, P. (2010). *Real analysis*. Prentice Hall.
Sahabandu, D., Moothedath, S., Allen, J., Bushnell, L., Lee, W., & Poovendran, R. (2019a). Stochastic dynamic information flow tracking game with reinforcement learning. In *International conference on decision and game theory for security* (pp. 417–438).
Sahabandu, D., Moothedath, S., Allen, J., Bushnell, L., Lee, W., & Poovendran, R. (2020). RL-ARNE: A reinforcement learning algorithm for computing average reward nash equilibrium of nonzero-sum stochastic games. *IEEE Transactions on Automatic Control* (conditionally accepted).
Sahabandu, D., Moothedath, S., Allen, J., Clark, A., Bushnell, L., Lee, W., et al. (2019b). A game theoretic approach for dynamic information flow tracking with conditional branching. In *American control conference* (pp. 2289–2296).
Sahabandu, D., Moothedath, S., Allen, J., Clark, A., Bushnell, L., Lee, W., et al. (2019c). Dynamic information flow tracking games for simultaneous detection of multiple attackers. In *IEEE conference on decision and control* (pp. 567–574).
Shapley, L. S. (1953). Stochastic games. *Proceedings of the National Academy of Sciences*, *39*(10), 1095–1100.
Siddiqui, S., Khan, M. S., Ferens, K., & Kinsner, W. (2016). Detecting advanced persistent threats using fractal dimension based machine learning classification. In *Proceedings of the 2016 ACM on international workshop on security and privacy analytics* (pp. 64–69).
Suh, G. E., Lee, J. W., Zhang, D., & Devadas, S. (2004). Secure program execution via dynamic information flow tracking. *ACM SIGPLAN Notices*, *39*(11), 85–96.
Tian, W., Ji, X.-P., Liu, W., Zhai, J., Liu, G., Dai, Y., et al. (2019). Honeypot game-theoretical model for defending against APT attacks with limited resources in cyber-physical systems. *ETRI Journal*, *41*(5), 585–598.
Vance, A. (2014). Flow based analysis of advanced persistent threats detecting targeted attacks in cloud computing. In *2014 first international scientific-practical conference problems of infocommunications science and technology* (pp. 173–176).
Vorobeychik, Y., An, B., & Tambe, M. (2012). Adversarial patrolling games. In *2012 AAAI spring symposium series*.
Wang, K., Perrault, A., Mate, A., & Tambe, M. (2020). Scalable game-focused learning of adversary models: Data-to-decisions in network security games. In *AAMAS* (pp. 1449–1457).
Watkins, B. (2014). The impact of cyber attacks on the private sector. *Briefing Paper, Association for International Affair*, *12*, 1–11.
Zhang, Y., Guo, Q., An, B., Tran-Thanh, L., & Jennings, N. R. (2019). Optimal interdiction of urban criminals with the aid of real-time information. In *Proceedings of the AAAI conference on artificial intelligence, Vol. 33* (pp. 1262–1269).
Zhu, Q., & Başar, T. (2011). Robust and resilient control design for cyber-physical systems with an application to power systems. In *IEEE decision and control and European control conference* (pp. 4066–4071).

Shana Moothedath is an Assistant Professor in the Department of Electrical and Computer Engineering at Iowa State University. She received her Ph.D. degree in Electrical Engineering from Indian Institute of Technology Bombay, India, in 2018. Her research interests include network security analysis, reinforcement learning, compressed sensing, and structural analysis of large-scale control systems.

Dinuka Sahabandu is a postdoctoral scholar at the Network Security Lab (NSL), Department of Electrical Engineering, at the University of Washington - Seattle. He received the Ph.D. degree from the Department of Electrical and Computer Engineering at the University of Washington - Seattle. He received the B.S. degree and M.S. degree in Electrical Engineering from the Washington State University - Pullman in 2013 and 2016, respectively. His research interests include game theory for network security and control of multi-agent systems.

Joey Allen is a Ph.D. candidate in the College of Computing at Georgia Institute of Technology, where he is studying information security under Dr. Wenke Lee. He received the B.S. degree and M.S. degree in Computer Engineering from the University of Tennessee - Knoxville in 2014 and 2016, respectively. His research interests include forensic analysis, mobile security, and information tracking.

Linda Bushnell (F '17) is a Research Professor in the Department of Electrical and Computer Engineering at the University of Washington (UW) - Seattle. She received her Ph.D. in Electrical Engineering from University of California - Berkeley in 1994, her M.A. in Mathematics from University of California - Berkeley in 1989, her M.S. and B.S. in Electrical Engineering from University of Connecticut - Storrs in 1987 and 1985 respectively. She also received her MBA from the UW Foster School of Business in 2010. Her research interests include networked control systems and secure-control. She is a Fellow of the IEEE and a Fellow of IFAC for contributions to the analysis and design of networked control systems. She is a recipient of the US Army Superior Civilian Service Award, NSF ADVANCE Fellowship, and IEEE Control Systems Society Distinguished Member Award. She is currently the Treasurer of the American Automatic Control Council, Member of the Technical Board for the International Federation on Automatic Control, Associate Editor for Automatica and for the IEEE Transactions on Control of Network Systems.

Wenke Lee is a Professor of Computer Science in the College of Computing at Georgia Institute of Technology. He also serves as the Director of the Georgia Tech Information Security Center (GTISC). He received his Ph.D. in Computer Science from Columbia University in 1999. His current research projects are in the areas of botnet detection and attribution, malware analysis, virtual machine monitoring, mobile phone security, and detection and mitigation of information manipulation on the Internet, with funding from NSF, DHS, DoD, and the industry. In 2006, Dr. Lee co-founded Damballa, Inc., a spin-off from his lab that focuses on botnet detection and mitigation.

Radha Poovendran (F'15) is a Professor in the Department of Electrical and Computer Engineering at the University of Washington (UW) - Seattle. He served as the Chair of the Electrical and Computer Engineering Department at UW for five years starting January 2015. He is the Director of the Network Security Lab (NSL) at UW. He is the Associate Director of Research of the UW Center for Excellence in Information Assurance Research and Education. He received the B.S. degree in Electrical Engineering and the M.S. degree in Electrical and Computer Engineering from the Indian Institute of Technology- Bombay and University of Michigan - Ann Arbor in 1988 and 1992, respectively. He received the Ph.D. degree in Electrical and Computer Engineering from the University of Maryland - College Park in 1999. His research interests are in the areas of wireless and sensor network security, control and security of cyber–physical systems, adversarial modeling, smart connected communities, control-security, games-security, and information theoretic security in the context of wireless mobile networks. He is a Fellow of the IEEE for his contributions to security in cyber–physical systems.

He is a recipient of the NSA LUCITE Rising Star Award (1999), National Science Foundation CAREER (2001), ARO YIP (2002), ONR YIP (2004), and PECASE (2005) for his research contributions to multi-user wireless security. He is also a recipient of the Outstanding Teaching Award and Outstanding Research Advisor Award from UW EE (2002), Graduate Mentor Award from Office of the Chancellor at University of California - San Diego (2006), and the University of Maryland ECE Distinguished Alumni Award (2016). He was co-author of award-winning papers including IEEE/IFIP William C. Carter Award Paper (2010) and WiOpt Best Paper Award (2012).