Learning Games for Defending Advanced Persistent Threats in Cyber Systems

Tianqing Zhu, Dayong Ye, Zishuo Cheng, Wanlei Zhou, and Philip S. Yu, *Fellow, IEEE*

***Abstract*—A cyber system may face to multiple attackers from diverse adversaries, who usually employ sophisticated techniques to both continuously steal sensitive data and avoid being detected by defense strategies. This continuous process is typically involved in an advanced persistent threat (APT). Since the game theory is an ideal mathematical model for investigating continuous decision making of competing players, it is broadly used to research the interaction between defenders and APT attackers. Although many researchers are now using the game theory to defend against APT attacks, most of the existing solutions are limited to single-defender, single-attacker scenarios. In the real world, threats by multiple attackers are not uncommon and multiple defenders can be put in place. Therefore, to overcome the limitation of the existing solutions, we develop a multiagent deep reinforcement learning (MADRL) method with a novel sampling approach. The MADRL method allows defenders to create strategies on the fly and share their experience with other defenders. To develop this method, we create a multidefender, multiattacker game model and analyze the equilibrium of this model. The results of a series of experiments demonstrate that, with MADRL, defenders can quickly learn efficient strategies against attackers.**

***Index Terms*—Advanced persistent threats (APTs), cyber system security, deep reinforcement learning, game theory.**

I. Introduction

CYBER systems and computer networks are vulnerable to various cyber attacks [1], [2], [3] One of the most severe cyber attacks is advanced persistent threats (APTs) [4], [5], where attackers utilize multiple methods, tools, and techniques to steal sensitive data from specific cyber systems. Unlike ordinary cyber attacks, APT attacks are usually long-term strategies designed to compromise the systems of a particular organization. Through continuous reconnaissance and investigation of the defense policy of the target, an APT attacker is very likely to avoid being detected [6]. Thus, finding an effective way to defend against APT attackers has been one of the major research problems in the cyber security domain [7].

APT attackers usually pursue their objectives repeatedly over an extended period of time. During this period, these attackers adapt to defenders' strategies by continuously changing their attack strategies. During APT attacks, defenders and attackers are competing players for the control power of target systems. Both of them repeatedly deploy defense/attack strategies over target systems. Therefore, the game theory is an effective model for investigating how to defend against APT attacks [8], [9]. As early as 2010, Alpcan and Basar [10] systematically discussed strategic decisions in network security games. Later, van Dijk et al. [11] modeled the interaction between a defender and an APT attacker as a stealthy takeover game, which covered the class of nontargeted APTs. The class of targeted APTs is addressed by more recent games, such as the cyber investment game [12] and cut-the-rope game [13]. However, most of the existing game models share one common limitation that they only accommodate one defender and one attacker. In the real world, cyber systems can be threatened by multiple attackers, and system designers can deploy multiple defenders [14]. Although some security game models consider multiple defenders [15], while others consider multiple attackers [16], no model considers both.

Even if the game theory is an excellent tool to analyze defense strategies and equilibria for APT attacks, analyzing a multidefender, multiattacker game model is difficult due to two challenges. First, multiplayer increases the difficulty of analyzing the equilibrium. As the equilibrium is affected by each defender's and attacker's strategy, analyzing the equilibrium is equivalent of searching strategy space. When the number of players increases, the strategy space increases exponentially. Second, deploying an efficient defense strategy with a large number of attackers and defenders can be difficult. Each defender's strategy is based on all other players' strategies. When the number of players increases, each defender has to consider more players' strategies. The difficulty of making an efficient defense strategy increases significantly. Reinforcement learning has been used to make defense strategies [8], [9], however, only in single-defender, single-attacker scenarios. In multidefender, multiattacker scenarios, the increase of the number of players implies the increase of the number of states, which will significantly reduce the learning speed of players. Thus, it is necessary

Manuscript received 19 April 2022; revised 24 July 2022; accepted 28 September 2022. Date of publication 19 October 2022; date of current version 17 March 2023. This work was supported in part by the National Science Foundation of China under Grant 61972366, and in part by NSF under Grant III-1763325, Grant III-1909323, Grant III-2106758, and Grant SaTC-1930941. This article was recommended by Associate Editor X. Ge. *(Corresponding author: Tianqing Zhu.)*

Tianqing Zhu is with the School of Computer Science, China University of Geosciences, Wuhan 430074, China, and also with the School of Computer Science, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: tianqing.e.zhu@gmail.com).

Dayong Ye and Zishuo Cheng are with the Centre for Cyber Security and Privacy and the School of Computer Science, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: dayong.ye@uts.edu.au; zishuo.cheng@uts.edu.au).

Wanlei Zhou is with the Institute of Data Science, City University of Macau, Macau, China (e-mail: wlzhou@cityu.mo).

Philip S. Yu is with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607 USA (e-mail: psyu@cs.uic.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2022.3211866.

Digital Object Identifier 10.1109/TSMC.2022.3211866

to develop a novel learning method for the multidefender, multiattacker scenario.

In this article, we design a multidefender, multiattacker game model. In this model, defenders cooperate with each other by sharing experience. When a defender creates strategies, she has to take the previous strategies of all the other defenders and attackers into account. By comparison, in single-defender, single-attacker models, the defender considers only the attacker's previous strategies. Thus, our multidefender, multiattacker model is much more complex than existing single-defender, single-attacker models. We first analyze the equilibrium of the game model in the multidefender, multiattacker scenario, and then adopt multiagent deep reinforcement learning (MADRL) for defenders to create strategies.

In regular reinforcement learning, the learned knowledge of a defender is stored in a table. If the number of players is large, the size of the table will be extremely huge and intractable. By comparison, in deep reinforcement learning, the learned knowledge of a defender is stored in a deep neural network. The size of the deep neural network is predefined and independent of the number of players. Hence, MADRL can handle multidefender, multiattacker scenarios efficiently. To the best of our knowledge, we are the first to address the multidefender, multiattacker cyber security game model by taking the advantage of MADRL. We find that MADRL can significantly increase the learning speed of defenders by enabling defenders to share their experience. In summary, this article makes three main contributions to the literature.

First, this research investigates the defense against APT attacks by considering multidefender, multiattacker scenarios, which are reflective of the real world but neglected in most related research. Second, we adopt MADRL to create strategies for defenders. As MADRL can efficiently deal with complex problems in high-dimensional environments, our method is more powerful than most of existing methods and can handle complex security problems. The adoption of MADRL to cyber security games opens up a new way for future researchers in this field. Third, to implement MADRL, we develop a novel sampling approach to select experience samples, which is more suitable for the security problem than existing sampling approaches. This sampling approach is based on the similarity of strategies, which encourages each defender to select experience samples derived from different strategies. Thus, the overfitting can be avoided.

II. Related Work

A great many efforts have been devoted to studying game theory for APT attacks [17], [18], [19]. Among these studies, incorporating reinforcement learning into APT game models is still a niche. In this section, our review focuses on reinforcement learning-based APT game models. For comprehensive surveys of the field, we refer readers to [20].

Xiao et al. [8] applied the prospect theory to model the interaction between a defender and an APT attacker. They used Prelec's probability weighting function to model the subjective decision making of the defender and attacker when they are unaware of their opponent's actions. They then designed a reinforcement learning algorithm for the defender to create defense strategies.

Later, Xiao et al. [9] extended their work [8] by adopting cumulative prospect theory to model a subjective defender and a subjective APT attacker. The model incorporates both the probability weighting distortion and the framing effect of the subjective defender and attacker. They developed a policy hill-climbing detection scheme to increase the policy uncertainty to fool the attacker. To accelerate the learning speed of the policy hill-climbing scheme, they also designed a hot-booting technique which uses learned experiences in similar scenarios to initialize the quality of strategies.

Min et al. [21] used a Colonel Blotto game to model the interaction between a defender and an APT attacker, where each player allocates resources over storage devices attempting to beat the other party. They developed a hot-booting resource allocation scheme, based on deep reinforcement learning, for the defender to create defense strategies.

Kamra et al. [22] proposed a model named deep fictitious play (DeepFP) to find the Nash equilibrium in continuous action spaces. DeepFP consists of a shared game model network and two best response networks, where one best response network belongs to the defender and the other to the attacker. Each player uses their best response network to select an action. Then, the game model network takes both players' actions as input and outputs the predicted rewards for both players. After that, the selected actions and the predicted rewards are used to update the game model network and best response networks.

Dunstatter et al. [23] presented a framework to derive game-theoretic strategies for assigning security alerts to security analysts. In the framework, they used fictitious play to find the Nash equilibrium by training a deep Nash Q-network. Unlike regular deep Q-networks (DQNs), in the proposed deep Nash Q-network, the actions of both the defender and the attacker are involved in the experience samples. This means that the optimization of the network takes both players into account.

Eghtesad et al. [24] proposed a partially observable Markov decision process (MDP) model for moving target defense (MTD). Then, they formulated the problem of finding adaptive MTD policies as finding the mixed-strategy Nash equilibrium of a game between the defender and the attacker. The problem is addressed using a deep Q-learning algorithm and a double oracle algorithm, where a player's actions are treated as part of the observation of the other player.

A. Other Related Research

In addition to the works reviewed above, there is also other research moderately related to ours. For example, Lv and Ren [25] studied a multiplayer nonlinear game using reinforcement learning by considering both zero-sum and nonzero-sum games. Their research was conducted in a general game setting with an assumption that player N cooperated with players 1 to $N-1$ and competed with player $N+1$. By comparison, our research is carried out in the cyber security domain without such an assumption. Zhang et al. [26] proposed a

demand-response game for energy trading in a blockchain-empowered system. Although their research also focuses on the game theory for resource allocation, it is different from ours in the problem domain: P2P energy markets versus cyber security systems. Zhang et al. [27] constructed a two-player zero-sum stochastic game in wireless networked control systems against denial-of-service (DoS) attacks by considering both the defense and attack strategies. Their research was conducted in a two-player setting and against DoS attacks. In contrast, our research is taken in a multiplayer setting and against APT attacks.

B. Discussion of Related Works

Most of the existing works focus on single-defender, single-attacker scenarios, overlooking scenarios with multiple defenders and attackers. As discussed in Section I, the research of a multidefender, multiattacker game model brings two challenging issues: 1) equilibrium analysis of the game model and 2) creating efficient strategies for defenders.

Equilibrium analysis in multiplayer games has been undertaken by game theoretic researchers [28], [29] but not been widely studied in the cyber security domain. The challenge here is that the results of research into general multiplayer game theory may not be applied directly to cyber security. This is because those results are derived from general game models, such as the prisoner's dilemma (PD) game, while cyber security games are typically based on specific game models, such as the Stackelberg game [15], [30] or Colonel Blotto game [21]. These specific game models have unique properties compared to general game models. For example, in the Stackelberg game, players make decisions alternatively instead of simultaneously in the PD game. In the Colonel Blotto game, players make decisions regarding resource distribution over a set of objects instead of selecting an individual action in the PD game.

The existing methods for creating defense strategies are developed on the basis of single-defender, single-attacker scenarios [8], [21]. They, however, might not be very efficient in multidefender, multiattacker scenarios due to the increased complexity. In this article, we propose a multidefender, multiattacker game model and analyze its equilibrium. We also propose an MADRL method for defenders to create efficient strategies by sharing experience with each other.

III. Cyber Security Game Model

We consider a network which consists of N hosts. These hosts are protected by M defenders and threatened by L attackers, as shown in Fig. 1. Here, the multiattacker scenario is used to simulate the situation that each individual attacker can make attack strategies autonomously given that these attackers are independent of each other though they may collude to achieve a common goal. The values of N, M, and L are independent of each other in general, but may be constrained to achieve specific results in theoretical analysis.

Each defender i has B_i defense resources, while each attacker j has C_j attacking resources. Resources in our work are abstract, which could represent the percentage of CPU power consumption with unit %, the amount of used memory with unit MB, the size of occupied hard-drive with unit MB/s, the bandwidth of occupied network with unit Mbps, and so on. Each host k has an importance value u_k which is based on the amount of data stored on host k.

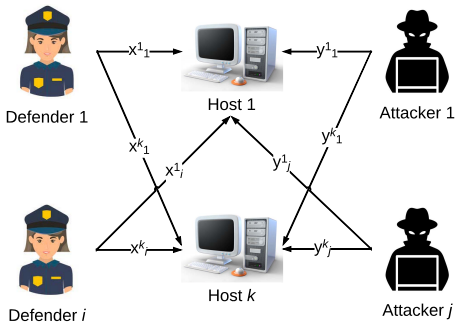

Fig. 1. Example of the cyber security game, where x_i^k is the number of defense resources allocated by defender i to host k and y_j^k is the number of attack resources allocated by attacker j to host k.

This cyber security game is modeled as the Colonel Blotto game [31], where two players simultaneously distribute forces across battlefields. Within each battlefield, the player that allocates the higher level of force wins. The rationale of using the Colonel Blotto game is that our research focuses on how defenders allocate defense resources over hosts to defeat attackers. This research problem perfectly matches the setting of the Colonel Blotto game model. In the cyber security game, we extend the model to the multidefender, multiattacker scenario. In each round t, the defenders and attackers take actions simultaneously by allocating their defense and attack resources, respectively, over the N hosts. Each player has an allocation strategy set, where defender i and attacker j's strategy sets are denoted as X_i and Y_j, respectively. Formally

$$X_i = \left\{ \boldsymbol{x}_i = \left(x_i^1, \ldots, x_i^N\right) | \forall k \in \{1, \ldots, N\} \right. \\ \left. 0 \le x_i^k \le B_i \wedge \sum_{1 \le k \le N} x_i^k \le B_i \right\} \tag{1}$$

and

$$Y_j = \left\{ \boldsymbol{y}_j = \left(y_j^1, \ldots, y_j^N\right) | \forall k \in \{1, \ldots, N\} \right. \\ \left. 0 \le y_j^k \le C_j \wedge \sum_{1 \le k \le N} y_j^k \le C_j \right\}. \tag{2}$$

A strategy set involves the available strategies that a player can take. In this article, we assume that a resource is indivisible and, thus, we consider only integer resources. Therefore, strategies are discrete. We use an example to explain strategies. Suppose there are a defender, an attacker and two hosts. Both the defender and attacker have two resources. For the defender, she has four available strategies: $X = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$. Each strategy represents a resource allocation over the hosts. For example, strategy (0, 1)

means allocating 0 resource to the first host while allocating 1 resource to the second host. Likewise, for the attacker, he also has four available strategies: $Y = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$.

For each host k, if $\sum_{1\le i\le M} x_i^k > \sum_{1\le j\le L} y_j^k$, the defender group wins and host k is safe. If $\sum_{1\le i\le M} x_i^k < \sum_{1\le j\le L} y_j^k$, the attacker group wins and host k is at risk. If $\sum_{1\le i\le M} x_i^k = \sum_{1\le j\le L} y_j^k$, a tie is established. Both the defender group and the attacker group have equal opportunity to control host k.

Defenders and attackers may take pure strategies: $(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M)$ and $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_L)$. In this situation, the defender group's utility is defined as follows:

$$U(\boldsymbol{x}, \boldsymbol{y}) = \sum_{1\le k\le N} u_k \cdot \operatorname{sgn}\left(\sum_{1\le i\le M} x_i^k - \sum_{1\le j\le L} y_j^k\right) \tag{3}$$

where $\operatorname{sgn}(a)$ is a sign function with $\operatorname{sgn}(a) = 1$ if $a > 0$, $\operatorname{sgn}(a) = -1$ if $a < 0$, and $\operatorname{sgn}(a) = 0$ if $a = 0$. Similarly, the attacker group's utility is defined as follows:

$$V(\boldsymbol{x}, \boldsymbol{y}) = \sum_{1\le k\le N} u_k \cdot \operatorname{sgn}\left(\sum_{1\le j\le L} y_j^k - \sum_{1\le i\le M} x_i^k\right). \tag{4}$$

For each defender and attacker, her and his utility is (U/M) and (V/L), respectively. In particular, when there is only one defender and one attacker, the defender's utility is

$$U(\boldsymbol{x}, \boldsymbol{y}) = \sum_{1\le k\le N} u_k \cdot \operatorname{sgn}\left(x^k - y^k\right) \tag{5}$$

while the attacker's utility is

$$V(\boldsymbol{x}, \boldsymbol{y}) = \sum_{1\le k\le N} u_k \cdot \operatorname{sgn}\left(y^k - x^k\right). \tag{6}$$

Defenders and attackers may also take mixed strategies: $(p(\boldsymbol{x}), q(\boldsymbol{y}))$ where $p(\boldsymbol{x}) = (p(\boldsymbol{x}_1), \ldots, p(\boldsymbol{x}_M))$ and $q(\boldsymbol{y}) = (q(\boldsymbol{y}_1), \ldots, q(\boldsymbol{y}_L))$, and each $p(\boldsymbol{x}_i)$ and $q(\boldsymbol{y}_j)$ is a probability distribution on strategy sets X_i and Y_j, respectively. In other words, a mixed strategy assigns a probability distribution over pure strategies that are the resource allocation schemes over the hosts with a given total resource consumption. Formally, the defender group's utility is defined as follows:

$$U(p(\boldsymbol{x}), q(\boldsymbol{y})) = \int_{\boldsymbol{x}\in X}\int_{\boldsymbol{y}\in Y} U(\boldsymbol{x}, \boldsymbol{y}) dp(\boldsymbol{x}) dq(\boldsymbol{y}) \tag{7}$$

where $X = X_1 \times \cdots \times X_M$ and $Y = Y_1 \times \cdots \times Y_L$. Similarly, the attacker group's utility is defined as follows:

$$V(p(\boldsymbol{x}), q(\boldsymbol{y})) = \int_{\boldsymbol{y}\in Y}\int_{\boldsymbol{x}\in X} V(\boldsymbol{x}, \boldsymbol{y}) dp(\boldsymbol{y}) dq(\boldsymbol{x}). \tag{8}$$

The aim of the defender group is to maximize its overall utility, while the aim of the attacker group is opposite.

In the following sections, we first address the first challenge by analyzing the equilibrium of defenders' and attackers' utility. Then, we address the second challenge by presenting the MADRL approach.

TABLE I
MEANING OF EACH NOTATION

Notations	Meaning
N	The number of hosts in the network
M/L	The number of defenders/attackers
B_i/C_j	The number of resources of defender i/attacker j
$\boldsymbol{x}_i/\boldsymbol{y}_j$	The pure strategy of defender i/attacker j
$p(\boldsymbol{x}_i)/q(\boldsymbol{y}_j)$	The mixed strategy of defender i/attacker j
U/V	The utility of the defender/attacker group
u_k	The importance value of host k
$\hat{u}$	The total importance value of all the N hosts

IV. ANALYSIS OF THE GAME MODEL

The commonly used notations are listed in Table I. We first provide a general conclusion that our game model can guarantee the cooperation of both defenders and attackers. This conclusion is important, as otherwise, players may take the strategies that increase only their own utility but harm the overall utility of the group. Then, we analyze the equilibrium of the game model in mixed strategies.

Theorem 1: The cyber security game model defined in Section III can guarantee the cooperation of both defenders and attackers.

Proof: In this model, each defender's and attacker's utility is set to (U/M) and (V/L), respectively. For each defender, the only way to increase her own utility is to increase the value of U, given that the value of M is fixed. As U is the overall utility of the defender group, any strategies, which can increase the value of U, are taken as cooperative strategies. Hence, all the defenders are cooperative. This conclusion also holds for attackers. ■

Note that uniformly distributing utility to members is not the only method to incentivize their cooperation. Another possible method is distributing utility based on each member's contribution, i.e., the member who uses more resources will be paid more. The choice of these utility distribution methods, however, does not affect the conclusion of Theorem 1.

When defenders and attackers take mixed strategies, we have the following conclusions.

Theorem 2: Let $\sum_{i=1}^{M} B_i = B$ and $\sum_{j=1}^{L} C_j = C$. If $B = C$ and both defenders and attackers uniformly allocate resources over hosts, then the game has a mixed strategy equilibrium, and the utility for both parties in this equilibrium is 0.

Proof: We prove this theorem from the defender side. For the attacker side, the proof is similar.

As $B = C$, the defender group and the attacker group have the same set of resource allocation strategies, i.e., actions. Let the number of actions be n. Given an $n \times n$ payoff matrix G, as this is a zero-sum game, the players' objective functions can be optimized using the minimax theorem. In particular, for the defender group, the optimization of their objective function is: $\operatorname{maxmin}(U) = \max_{p\in\Delta[n]} \min_{y\in[n]} \sum_{i=1}^{n} p_i \cdot G(i, y)$, while for the attacker group, it is: $\operatorname{minmax}(U) = \min_{q\in\Delta[n]} \max_{x\in[n]} \sum_{j=1}^{n} q_j \cdot G(x, j)$, where $\Delta[n]$ denotes the set of probability distributions over $[n] = \{1, \ldots, n\}$, p_i and q_j are the probabilities with which the defender group selects action i and the attacker group picks up action j, respectively, and x and y are the indices of the row and column of the payoff matrix G, respectively. Given that the attacker

group will always play the action that reaches the lowest utility to the defender, thus, to maximize the defender group's utility, defenders should play to equalize the utility of the actions, which is formalized as: $\sum_{i=1}^{n} p_i \cdot G(i,1) = \cdots = \sum_{i=1}^{n} p_i \cdot G(i,n) = u$, where u denotes the expected equal utility. These equations can be reformatted as an equation system

$$\begin{cases} p_1 \cdot G(1,1) + \cdots + p_n \cdot G(n,1) = u \\ \cdots \\ p_1 \cdot G(1,n) + \cdots + p_n \cdot G(n,n) = u. \end{cases}$$

When summing these equations up, we have $p_1 \sum_{i=1}^{n} G(1,i) + \cdots + p_n \sum_{i=1}^{n} G(n,i) = u \cdot n$. When the defender group uses the uniform strategy, then $p_1 = \cdots = p_n = (1/n)$. The above equation becomes $(1/n)\sum_{i=1}^{n} G(1,i) + \cdots + (1/n)\sum_{i=1}^{n} G(n,i) = (1/n)\sum_{j=1}^{n}\sum_{i=1}^{n} G(j,i) = u \cdot n$. Note that $\sum_{j=1}^{n}\sum_{i=1}^{n} G(j,i)$ means summing up all the elements in the payoff matrix G. As this game is zero-sum and the defender group and the attacker group have the same action set, based on the definition of utility functions, we have $G(j,i) = -G(i,j)$ and $G(i,i) = 0$. Thus, $\sum_{j=1}^{n}\sum_{i=1}^{n} G(j,i) = 0$. With this result, we have $u = 0$, meaning that when using the uniform strategy, the defender group's expected utility is 0. By symmetry, this conclusion also applies to the attacker group. ■

Theorem 2 has analyzed the situation that the defenders and the attackers have the same total number of resources. We now discuss the situation that they have different number of resources.

Theorem 3: Given $(N/2) \le B \le N \le C$, where B and C are the number of resources of each defender and attacker, respectively, if defenders use a mixed strategy $p(\boldsymbol{x})$, where for each $k \in \{1,\ldots,N\}$ and $i \in \{1,\ldots,M\}$, x_i^k is an integer uniformly drawn from $[0, (2B/N)]$, the defender group's utility $U(p(\boldsymbol{x}), \boldsymbol{y}) \le 0$ for any strategy $\boldsymbol{y}$ of the attackers that for each $k \in \{1,\ldots,N\}$ and $j \in \{1,\ldots,L\}$, $y_j^k \ge 1$.

Proof: For simplicity, we use $\sum_k$ for $\sum_{1\le k\le N}$, $\sum_i$ for $\sum_{1\le i\le M}$ and $\sum_j$ for $\sum_{1\le j\le L}$, respectively. Given the attackers' strategy $\boldsymbol{y}$, the defender group's utility is

$$\begin{aligned} U(p(\boldsymbol{x}), \boldsymbol{y}) &= \int_{\boldsymbol{x}\in X} U(\boldsymbol{x}, \boldsymbol{y}) dp(\boldsymbol{x}) \\ &= \int_{\boldsymbol{x}\in X} \sum_k u_k \cdot \operatorname{sgn}\left(\sum_i x_i^k - \sum_j y_j^k\right) dp(\boldsymbol{x}). \end{aligned}$$

Since each x_i^k is uniformly drawn from $(2B/N)$, we have $\sum_i x_i^k \le (2BM/N)$, where M is the number of defenders. Moreover, $p(\mathbf{x})$ is the cumulative distribution function of $\mathbf{x}$, so that

$$dp(\mathbf{x}) = d\frac{\sum_i x_i^k}{2BM/N}.$$

Letting $z = \sum_i x_i^k$, then the above integral can be converted to

$$U(p(\boldsymbol{x}), \boldsymbol{y}) = \sum_k u_k \int_{z=0}^{\frac{2BM}{N}} \operatorname{sgn}\left(z - \sum_j y_j^k\right) \frac{N}{2BM} dz.$$

On the one hand, if $\sum_j y_j^k > (2BM/N)$, we have

$$\int_{z=0}^{\frac{2BM}{N}} \operatorname{sgn}\left(z - \sum_j y_j^k\right) \frac{N}{2BM} dz = \int_{z=0}^{\frac{2BM}{N}} -\frac{N}{2BM} dz = -1.$$

On the other hand, if $\sum_j y_j^k \le (2BM/N)$, we have

$$\begin{aligned} &\int_{z=0}^{\frac{2BM}{N}} \operatorname{sgn}\left(z - \sum_j y_j^k\right) \frac{N}{2BM} dz \\ &= \int_{z=0}^{\sum_j y_j^k} -\frac{N}{2BM} dz + \int_{\sum_j y_j^k}^{\frac{2BM}{N}} \frac{N}{2BM} dz \\ &= -\frac{N}{2BM}\sum_j y_j^k + \frac{N}{2BM}\left(\frac{2BM}{N} - \sum_j y_j^k\right) \\ &= \left(\frac{BM}{N} - \sum_j y_j^k\right)\frac{N}{BM}. \end{aligned}$$

In summary, if $\sum_j y_j^k > (2BM/N)$, we have

$$U(p(\boldsymbol{x}), \boldsymbol{y}) = -\sum_k u_k = -\hat{u} < 0.$$

If $\sum_j y_j^k \le (2BM/N)$, we have

$$U(p(\boldsymbol{x}), \boldsymbol{y}) = \sum_k u_k \left(\frac{BM}{N} - \sum_j y_j^k\right)\frac{N}{BM}.$$

As $B \le (N/M)$, then $(BM/N) \le 1$ and, thus, $(BM/N) - \sum_j y_j^k \le 0$ which means that $U(p(\boldsymbol{x}), \boldsymbol{y}) \le 0$. Therefore, in both scenarios, $U(p(\boldsymbol{x}), \boldsymbol{y}) \le 0$. ■

Theorem 3 states that if any defender's resources are insufficient to cover all the hosts and the defenders use mixed strategies, while each attacker's resources are sufficient to cover all the host, then as long as each attacker allocates at least one resource to each host, the defender group cannot receive positive utility irrespective of the utility of each host. By symmetry, attackers have similar conclusions which are not given here for simplicity.

In this work, the cooperation among defenders is interpreted as the share of learning experience (refer to Section V). The experience sharing is not reflected in the theoretical analysis, because it is not involved in the game model. A possible way to introduce complex dynamics into the game model is to enable members to transfer resources between one another. In this situation, the action space of each member may change in each game round, because the number of resources that each member has may change in each round. We will briefly discuss the equilibrium in this situation and leave the deep investigation as our future work.

Formally, let X and Y be the sets of all potentially attainable strategies of a defender and an attacker, respectively. Also, let $X^t \subset X$ and $Y^t \subset Y$ be the sets of the defender's and the attacker's strategies, respectively, in round t. Correspondingly, let $G^t \in \mathbb{R}^{|X^t|\times|Y^t|}$ be the payoff matrix in round t, where $|X^t|$ and $|Y^t|$ denote the numbers of strategies of the defender and the attacker, respectively, in round t. Let Δ_d^t and Δ_a^t be the

sets of probability distributions over the defender's and the attacker's available strategies, respectively, in round t. Then, if the defender uses a probability distribution $x \in \Delta_d^t$ and the attacker uses a probability distribution $y \in \Delta_a^t$, the defender's utility is xG^ty. The defender aims to maximize the quantity of xG^ty, while the attacker has the opposite aim. This is a minimax problem. The solution to this problem is an equilibrium of the game [32]. However, since X^t and Y^t may change in each round, the payoff matrix G^t also changes in each round. Accordingly, the solution changes in each round. Hence, it is intractable to give a general form of the solution.

V. Multiagent Deep Reinforcement Learning Method

In the above game model, we analyze that in which situations, the defenders and attackers can achieve an equilibrium, which demonstrate the lower bound of defenders' utility. However, the game model discovers only general properties of the interaction between defenders and attackers, but does not offer a solution for defenders to defeat attackers. In this section, we propose an MADRL method for defenders to make efficient strategies to defeat attackers.

A. Preliminary of Multiagent Deep Reinforcement Learning

Reinforcement learning is usually used to solve sequential decision-making problems, where an agent learns through a trial-and-error manner in an environment. Formally, the sequential decision-making process can be modeled as an MDP [33], which is depicted as a tuple $\langle S, A, T, R\rangle$. In this tuple, S is the set of states, A is a set of actions available to the agent, T is the transition function, and R is the reward function.

At each step, the agent observes the state $s \in S$ of the environment, and selects an action $a \in A$ based on its policy π. After performing the action, the agent receives a real-valued reward r, and the environment changes to a new state $s' \in S$. The agent then updates its policy π based on the reward r and the new state s'. The policy π can be represented as a Q-function $Q(s, a)$ which estimates the reward that the agent will earn in state s by performing action a. Through this way, the agent can gradually accumulate knowledge and improve its policy to maximize its total long-term expected reward.

Regular reinforcement learning cannot deal with complex problems, e.g., a large number of states and/or actions. DQN [34] is thus introduced into reinforcement learning to form deep reinforcement learning, which can handle complex problems. In regular reinforcement learning, an agent maintains and regularly updates a Q-table, and each entry in the Q-table is the Q-value of a state, action pair: $Q(s, a)$. However, when the number of states and/or actions is large, the Q-table will become extremely large, which is hardly to be maintained and updated in a timely manner. In contrast, in deep reinforcement learning, a neural network is used to approximate the Q-table. The input of the neural network could be a state, action pair, and the output is the Q-value of this pair: $Q(s, a)$. Since maintaining a neural network is much less expensive than maintaining a large Q-table, deep reinforcement learning can handle complex problems.

Single agent deep reinforcement learning can be extended to multiagent environments to form MADRL by letting each agent using deep reinforcement learning. In MADRL, state transitions are based on the joint actions of all the agents. Each agent's reward is also based on the joint actions of all the agents. Moreover, agents may interact with each other to share knowledge so as to accelerate their learning speed [35], [36], [37].

B. Details of the MADRL Method

1) Novel Strategy-Based Sampling Approach: In our MADRL method, a novel strategy-based sampling approach is developed to select samples from the experience replay set (line 14 of Algorithm 1). Popular sampling approaches in deep reinforcement learning include: random sampling [34], gradient-based sampling [38], and temporal difference error-based sampling [39], [40]. These approaches are not designed for security games. In our security game, each defender may have a large number of available actions, i.e., strategies. During sampling, to avoid overfitting, defenders should avoid selecting the samples which have similar strategies. Thus, the similarity between strategies must be taken into consideration.

Our strategy-based sampling approach uses the Euclidean distance to measure the similarity between strategies. First, the defender calculates the "average strategy," $\bar{\boldsymbol{x}}$, using the samples in the experience set D via

$$\bar{\boldsymbol{x}} = \frac{1}{|D|} \sum_{1 \le j \le |D|} \boldsymbol{x}^j. \tag{9}$$

Next, for each $\boldsymbol{x}^j$, the defender calculates the Euclidean distance between $\boldsymbol{x}^j$ and $\bar{\boldsymbol{x}}$ via

$$\text{dist}\left(\boldsymbol{x}^j, \bar{\boldsymbol{x}}\right) = ||\boldsymbol{x}^j - \bar{\boldsymbol{x}}||_2. \tag{10}$$

Finally, the defender ranks the $|D|$ samples in the decreasing order with $\text{dist}(\boldsymbol{x}^j, \bar{\boldsymbol{x}})$, and selects the top m samples. The reason why the defender selects samples based on the samples' ranking positions rather than their probabilities defined by the proportion of the distance values is that prioritization on experience using rank-based information has been proven to be robust and efficient [38], [39].

2) Notations Used in the MADRL Method: Each defender's strategy in current round t is based on both the defenders' and the attackers' strategies in the last T rounds. Thus, for defender i, her state in round t is defined as $s_i^t = \{o_i^{t-T}, \ldots, o_i^{t-1}\}$, where $o_i^{t-j} = (\boldsymbol{x}_1^{t-j}, \ldots, \boldsymbol{x}_M^{t-j}; \boldsymbol{y}_1^{t-j}, \ldots, \boldsymbol{y}_L^{t-j})$ and $1 \le j \le T$. Here, a state consists of a set of observations. Each observation includes the strategies of all the defenders and attackers in a round. If the environment is partially observable, for defender i, we can remove the strategies of those defenders and attackers from her observation, which cannot be observed by defender i.

Defender i's actions are defined as her strategies: $a_i = \boldsymbol{x}_i$, and we will use a_i and $\boldsymbol{x}_i$ interchangeably. Since strategies are discrete and actions stand for strategies in the deep Q-learning method, actions are also discrete. Defender i's

Algorithm 1: Deep Q-Learning Algorithm for the Single-Defender, Single-Attacker Scenario

1 **Input**: the defender's state, weight $\boldsymbol{\theta}$ of the deep Q-network (DQN), and experience replay set D;
2 **Output**: a trained DQN;
3 Initialize $\boldsymbol{\theta}$ randomly;
4 Initialize Q-values of each action randomly;
5 Set $D = \emptyset$;
6 **for** $t = 1$ *to the end of the game* **do**
7 With probability ϵ, randomly select a strategy $\boldsymbol{x}^t$, otherwise select $\boldsymbol{x}^t = argMax_{\boldsymbol{x}} Q(s^t, \boldsymbol{x}; \boldsymbol{\theta})$;
8 Perform strategy $\boldsymbol{x}^t$;
9 Receive reward r^t and observation o^{t+1};
10 Use $(s^t, \boldsymbol{x}^t)$ as input to the DQN;
11 Obtain output vector $\boldsymbol{Q}(s^t)$ from the DQN;
12 Preprocess state s^{t+1}: $s^{t+1} \leftarrow s^t \cup \{o^{t+1}\} - \{o^{t-T}\}$;
13 Put sample $(s^t, \boldsymbol{x}^t, r^t, s^{t+1})$ into D;
14 Select m samples from D;
15 Calculate target Q-values of these samples: $Q_j \leftarrow r^j + \gamma \cdot max_{\boldsymbol{x}} Q(s^{j+1}, \boldsymbol{x}; \boldsymbol{\theta})$, where $1 \le j \le m$;
16 According to loss function: $(1/m) \sum_{1 \le j \le m} [Q_j - Q(s^j, \boldsymbol{x}^j; \boldsymbol{\theta})]^2$, use gradient descent to update $\boldsymbol{\theta}$;

Algorithm 2: Deep Q-Learning Algorithm for the Multidefender, Multiattacker Scenario

1 Take defender i as an example;
2 **Input**: defender i's state, weight $\boldsymbol{\theta}_i$ of the deep Q-network (DQN), and experience replay set D_i;
3 **Output**: a trained DQN;
4 Initialize $\boldsymbol{\theta}$ randomly;
5 Initialize Q-values of each action randomly;
6 Set $D_i = \emptyset$;
7 **for** $t = 1$ *to the end of the game* **do**
8 With probability ϵ, randomly select a strategy $\boldsymbol{x}_i^t$, otherwise select $\boldsymbol{x}_i^t = argMax_{\boldsymbol{x}} Q(s_i^t, \boldsymbol{x}_i; \boldsymbol{\theta}_i)$;
9 Perform strategy $\boldsymbol{x}_i^t$;
10 Receive reward r_i^t and observation o_i^{t+1};
11 Use $(s_i^t, \boldsymbol{x}_i^t)$ as input to the DQN;
12 Obtain output vector $\boldsymbol{Q}(s_i^t)$ from the DQN;
13 Preprocess state s_i^{t+1}: $s_i^{t+1} \leftarrow s_i^t \cup \{o_i^{t+1}\} - \{o_i^{t-T}\}$;
14 Put sample $(s_i^t, \boldsymbol{x}_i^t, r_i^t, s_i^{t+1})$ into D_i;
15 Select m samples from $D_1, ..., D_M$;
16 Calculate target Q-values of these samples: $Q_j \leftarrow r_i^j + \gamma \cdot max_{\boldsymbol{x}} Q(s_i^{j+1}, \boldsymbol{x}; \boldsymbol{\theta}_i)$, where $1 \le j \le m$;
17 According to loss function: $(1/m) \sum_{1 \le j \le m} [Q_j - Q(s_i^j, \boldsymbol{x}_i^j; \boldsymbol{\theta}_i)]^2$, use gradient descent to update $\boldsymbol{\theta}_i$;

reward in round t is defined as the ratio between her received utility and the number of her used resources in round t: $r_i^t = ([U^t]/[M \cdot \sum_{1 \le k \le N} x_i^k])$. Particularly, when we have only one defender and one attacker, the defender's state in round t is defined as $s^t = \{o^{t-T}, \ldots, o^{t-1}\}$, where $o^{t-j} = (\boldsymbol{x}^{t-j}, \boldsymbol{y}^{t-j})$ and $1 \le j \le T$.

According to the deep Q-learning method [34], we first propose Algorithm 1 to handle the single-defender, single-attacker scenario, and then extend it to Algorithm 2 to accommodate the multidefender, multiattacker scenario.

3) Single-Defender, Single-Attacker: In Algorithm 1, in each round of the game, the defender first selects a strategy using ϵ-greedy approach (line 7). The probability ϵ is a hyperparameter which is typically tuned by the users during experiments to achieve the ideal results. The defender then performs the selected strategy, and receives a reward and an observation (lines 8 and 9). Next, the defender updates the Q-value vector in the current state using the DQN (lines 10 and 11). After that, the defender appends the new observation to the state and removes the oldest observation from the state (line 12). The defender puts the sample with the updated state into the experience replay set, and selects m samples from the set (lines 13 and 14). Finally, the defender uses the selected samples to update the weights of the DQN by minimizing the loss function defined by the difference between target Q-values and learned Q-values (lines 15 and 16).

4) Multidefender, Multiattacker: The major part of Algorithm 2 is the same as Algorithm 1. The main difference between them includes two parts. The first part is the input of the DQN (line 1 in Algorithm 2). The input is a state of a defender. For the single-defender, single-attacker case in Algorithm 1, a state of the defender consists of the strategies that the defender and attacker used in the last T rounds. By comparison, for the multidefender, multiattacker case in Algorithm 2, a state of a defender consists of the strategies that all the defenders and attackers used in the last T rounds. Thus, the dimension and complexity of states in Algorithm 2 are much higher than in Algorithm 1. The aim of Algorithm 2 is to let each defender learn a mapping from the attackers' previous strategies to a defense strategy, no matter how the attackers' strategies are created. Thus, even if smart attackers may change their strategies based on defenders' previous strategies, the defenders can still obtain efficient defense strategies, as long as the defenders can observe the attackers' strategies. The effectiveness of the proposed method will be shown in the experimental part, where each attacker considers not only the defenders' previous strategies but also each host's importance value.

The second difference is the sampling process (line 15 in Algorithm 2). For the single-defender, single-attacker case in Algorithm 1, the defender selects samples only from her own experience replay sample set. By comparison, for the multidefender, multiattacker case in Algorithm 2, a defender selects samples from all the defenders' experience replay sample sets. By sharing experience samples, defenders can learn much faster than using only their own experience samples [41], [42].

A potential issue in Algorithm 2 is that if defenders have different action spaces, i.e., different amounts of resources, a defender's experience samples may not be used by another defender. This issue can be avoided in Algorithm 2. Each experience sample includes four tuples, $(s^t, \mathbf{x}^t, r^t, s^{(t+1)})$. When a

defender is selecting samples from other defenders, she will check the strategy $\mathbf{x}^t$ in each sample. If the strategy is not in her action space, this sample will be disregarded.

C. Analysis of the Method

1) Complexity Analysis:

Theorem 4: The time complexity of Algorithm 2 is $O(t \cdot (W + |D|))$, where t is the total number of game rounds, W is the number of strategies available to defender i, and $|D|$ is the sum of the sizes of all defenders' experience replay sets: $|D| = \sum_{1 \le i \le M} |D_i|$.

Proof: In line 8, defender i, with probability $1-\epsilon$, traverses her strategy set to select the strategy which has the highest Q-value. Moreover, in line 15, to calculate the average strategy, the defender has to traverse her experience replay set D. In lines 16 and 17, the complexity of the DQN update step is based on the number of samples m drawn from experience replay set D_i. Thus, the time complexity of the three steps is $O((1-\epsilon)W + |D_i| + 2m)$, where D_i is the experience replay set of defender i. However, as defender i can access other defenders' experience replay sets, m may be larger than $|D_i|$. In the extreme case, $m = |D| = \sum_{1 \le i \le M} |D_i|$, i.e., that defender i selects all the samples from all the defenders' experience replay sets. As $1-\epsilon$ is a constant and $m = |D| > |D_i|$ in the extreme case, the time complexity can be rewritten as $O(W + 2|D|) = O(W + |D|)$. Finally, as the game is played t rounds, the overall time complexity is $O(t \cdot (W + |D|))$. ■

The time complexity of Algorithm 2 scales linearly with the number of strategies and the size of experience replay set of each defender. The number of strategies of a defender is based on the number of hosts N and the number of the defender's resources B. Therefore, the time complexity is based on: 1) the number of hosts N; 2) the number of each defender's resources B_i; and 3) the size of each defender's experience replay set D_i. Among the three factors, the most important one is N, as N determines the scalability of our method.

2) Convergence Analysis: As mentioned in Section I, deep reinforcement learning algorithms approximate a Q-table with a neural network. This approximation, however, intrinsically introduces a statistical error. This error is caused by the bias of the ReLU activation function and finite sample size in deep Q-learning. Due to the nonlinearity of a deep neural network, the theoretical analysis of deep Q-learning tends to be intractable, as it involves solving a highly nonconvex statistical optimization problem. This statistical error has an impact on the theoretical convergence. Although it is infeasible to rigorously prove the convergence to an optimum, Fan et al. [43] have proven that a properly designed deep Q-learning algorithm can obtain a sequence of Q-networks that converges to the optimal action-value function with the statistical error as the error bound.[1] Motivated by [43], we first analyze the convergence of our algorithm and then discuss the equilibrium.

[1]Note that this conclusion may not hold for regular reinforcement learning algorithms which do not use deep neural networks.

Given an MDP model (S, A, P, R, γ), where S is the set of states, A is the set of actions, $P : S \times A \to S$ is the transition function denoting the probability with which a new state is reached if an action is performed in a given state, $R : S \times A \to R$ is the reward function indicating the immediate reward received by the player if an action is taken in a given state, and $\gamma \in (0, 1)$ is the discount factor. For a given MDP, a policy $\pi : S \to A$, made by the player, maps any state $s \in S$ to a probability distribution $\pi(\cdot|s)$ over A. Then, the corresponding state–action-value function $Q^\pi : S \times A \to R$ is defined as the cumulative discounted reward received by the player taking actions based on policy π, formally, $Q^\pi(s, a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \cdot R_t | S_0 = s, A_0 = a]$. As the aim of the player is to maximize her cumulative reward, the optimal state–action-value function can be given as: $Q^*(s, a) = \max_\pi Q^\pi(s, a)$. Now, we define the Bellman optimality operator T as: $(TQ)(s, a) = r(s, a) + \gamma \cdot \mathbb{E}[\max_{a' \in A} Q(S', a') | S' \sim P(\cdot|s, a)]$. Then, we have the Bellman optimality equation $TQ^* = Q^*$. It has been verified [33] that T is γ-contractive with respect to the supremum norm over $S \times A$. This means that for any two functions Q and Q', it holds that $|(|TQ - TQ'|)|_\infty \le \gamma \cdot |(|Q - Q'|)|_\infty$. Then, let $Q_k = TQ(k-1)$ for all $k \ge 1$, it holds that $|(|Q_k - Q^*|)|_\infty \le \gamma^k \cdot |(|Q_0 - Q^*|)|_\infty$. Thus, when $k \to \infty$, $\{Q_k\}_{k \ge 0}$ converges to the optimal state–action-value function Q^*.

By applying the above analysis to our algorithm, we can find that the DQN is used to approximate TQ^* by minimizing the loss function: $L(\theta) = (1/m) \sum_{i=1}^{m} [Q_i - Q_\theta(s_i, a_i)]^2$, where θ denotes the parameter of the DQN, m is the batch size, and $Q_i = r_i + \gamma \cdot \max_\pi (a \in A) Q_\theta(s_i', a)$ is the target for iteration i. Let us denote $Q_{k+1} = \hat{T}_k Q_k$, where $\hat{T}_k$ is an approximation of the Bellman optimality operator T learned from the training samples in the kth training round. Strictly defining $\hat{T}_k$ is intractable, because the update of Q_{k+1} is based on the update of the parameters in a highly complex DQN. Since the update of parameters uses the stochastic gradient descent, $\hat{T}_k$ is based on the gradient of the loss function $\nabla L(\theta)$ which aims to minimize the difference between the expected Q-value and the actual Q-value. During learning, we start from the initial value Q_0, collect the samples $\{(s_i, a_i, r_i, s_i')\}_{i \in [m]}$ and learn the map $\hat{T}_1$ and receives $Q_1 = \hat{T}_1 Q_0$; then collect a new batch of samples to learn a new map $\hat{T}_2$ and obtains $Q_2 = \hat{T}_2 Q_1$, and so forth. The final value becomes $Q_k = \hat{T}_k \cdots \hat{T}_1 Q_0$, which resembles the update process as the regular reinforcement learning. The use of gradient descent during learning can guarantee the convergence of the algorithm [44].

The above analysis indicates that the algorithm can converge. However, the convergence limit may not be an equilibrium. This is because the learning algorithm is designed to maximize the defender's utility, i.e., beating the attacker, rather than finding an equilibrium. To make the algorithm converge to an equilibrium, the loss function needs to be revised. Specifically, let us reformat the state–action-value function as: $Q^*(s, a, b) = \max_\pi \min_\upsilon Q^{\pi, \upsilon}(s, a, b)$, where a and b are the actions of the defender and attacker, respectively; π and υ are the policies of the defender and attacker, respectively. Then, the equilibrium joint policy can be defined as: $[\pi(\cdot|s), \upsilon(\cdot|s)] = \operatorname{argmax}_{\pi \in P(A)} \operatorname{argmin}_{\upsilon \in P(B)} \mathbb{E}_{a \sim \pi, b \sim \upsilon}[Q(s, a, b)]$. Hence, the

expected Q-value is revised as: $Q_i = r_i + \gamma \cdot \max_{a\in A} \min_{b\in B} Q_\theta(s_i', a, b)$, and also the loss function, $L(\theta) = (1/m)\sum_{i=1}^{m}[Q_i - Q_\theta(s_i, a_i)]^2$, will be revised accordingly. This revised expected Q-value reveals that the defender takes the attacker's actions into consideration and assumes that the attacker will use the actions to minimize the defender's utility. Then, if both players use the same loss function, the gradient descent will lead both of them to an equilibrium. This can be explained using the von Neumann's minimax theorem: $Q^*(s, a, b) = \max_\pi \min_\upsilon Q^{\pi,\upsilon}(s, a, b) = \min_\upsilon \max_\pi Q^{\upsilon,\pi}(s, a, b)$. This equation means when both players use the same loss function, they have the same optimization aim to reach Q^*, an equilibrium. The actions, i.e., resource allocation strategies, taken in this equilibrium, are based on the game setting, e.g., the number of resources of each player.

VI. Experiment and Analysis

We evaluate the proposed method, MADRL, in comparison with two baseline methods: 1) deep reinforcement learning (DRL) following [21] and 2) regular reinforcement learning (RL) following [9]. In the DRL method, each defender independently used a DRL algorithm to determine her strategies, so this comparison effectively assesses the impact of defenders sharing versus not sharing experience. In the RL method, each defender independently uses a regular Q-learning algorithm to determine her strategies. Therefore, this comparison effectively assesses the impact of using neural networks versus not using.

The average utility of defenders defined in (3) and (7) is used as the metric to evaluate each of the three methods. The parameter values are set as follows: learning rate $\alpha = 0.1$, discount factor $\gamma = 0.8$, greedy parameter $\epsilon = 0.8$, and sampling size $m = 3$. These parameter values were experimentally chosen to yield good results. Moreover, the DQN adopted in our method is a four-layer fully connected neural network. The input layer is the state of a defender as described in Section V-B. The output layer consists of the Q-values of the defender's each strategy. Each of the two hidden layers has 1000 nodes.

An attacker's strategy is based on hosts' importance values and the defender's previous strategies. To create a strategy, the attacker first computes the average number of resources allocated by the defender to each host in the previous t rounds. Then, the attacker ranks these hosts based on their importance values, and allocates his resources to the hosts with high importance values in priority. The amount of resources allocated to each host is based on the average number of resources allocated by the defender to each host in the previous t rounds. As an example, suppose that a network has two hosts, one defender and one attacker. The importance value of host 1 is greater than host 2, i.e., $u_1 > u_2$. Both the defender and the attacker have five resources. In the previous t rounds, the defender allocated three resources on host 1, on average, and two resources on host 2, on average. In the $t+1$ round, the attacker has known that $u_1 > u_2$ and the average amount of resources allocated by the defender to host 1 and host 2 is 3 and 2, respectively. As $u_1 > u_2$, the attacker will allocate his resources on host 1 in priority. To defeat the defender on host 1 with a moderate chance, the attacker should allocate more than three resources to host 1. Therefore, the attacker allocates four resources to host 1 and the remaining one resource to host 2.

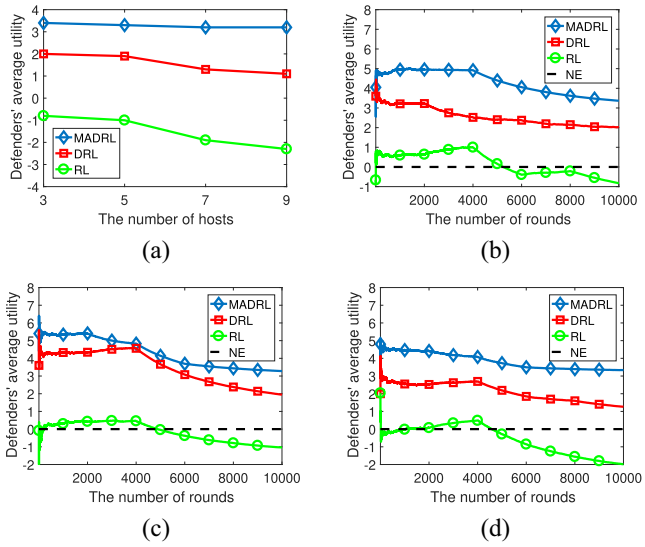

Fig. 2. Performance of the three methods in Situation 1. (a) Defenders' average utility with different number of hosts. (b) Defenders' average utility when hosts = 3. (c) Defenders' average utility when hosts = 5. (d) Defenders' average utility when hosts = 7.

Experiments are conducted in the following five situations. Situations 1–4 simulate certain environments, while Situation 5 simulates an uncertain environment. Both of the certain and uncertain environments can accommodate APT attacks [21], [45].

Situation 1: A varying number of hosts to evaluate scalability.

Situation 2: A varying number of attackers to evaluate robustness.

Situation 3: A varying average number of resources for each defender to evaluate adaptability. An "average" number means that a quantity is uniformly selected from a small range whose average value is that average number. For example, when the average number of resources of each defender is set to 6, this means that each defender's resource amount is uniformly selected from the range [5, 7] whose average value is 6.

Situation 4: Dynamically changing the importance values of hosts to evaluate suitability to dynamic environments.

Situation 5: Changing the game model to "probabilistic winning." for example, a defender allocates three resources on a host while an attacker allocates two resources on the same host; then the defender has probability (3/5) to win while the attacker has probability (2/5) to win.

A. Experimental Results

1) Situation 1: Fig. 2 shows how the methods perform when the number of hosts is varied from 3 to 9. The number of both defenders and attackers is set to 3. The average number of resources of each defender and attacker is set to 6, where the number of each defender's resource is uniformly drawn

from [5, 7]. The average utility of each host is set to 2, where the utility of each host is uniformly selected from [1, 3].

Fig. 2(a) shows that as the number of hosts increases, the average utility of the defenders remains almost steady with our MADRL method, while it gradually decreases with the DRL and RL methods. By increasing the number of hosts, the number of states and available strategies to each defender significantly increases. The RL method is not able to handle this increase in complexity. The DRL method is better than the RL method, because it uses a neural network to compress the learning state space, which accelerates learning speed. However, the MADRL defenders share their experience, whereas the DRL defenders do not, and this gives MADRL the edge on this assessment.

Fig. 2(b)–(d) shows how the average utility of the defenders changes for each method as the game progresses. The MADRL method enables defenders to obtain more utility than the DRL method under different numbers of hosts. Since the defenders with the DRL method do not share experience, they may instead prefer to deploy resources over high-utility hosts. However, these deployments can mean that resources are used wastefully, leaves the way low-utility hosts taken by attackers. Many low-utility hosts may be worth more than only a few high-utility hosts, so the defenders' overall utility may suffer. The RL method is clearly less stable than the other two methods. Its defenders have a long learning period with a large number of states and strategies, because each defender chooses its strategies randomly. This also accounts for the high fluctuations in average utility for the defenders with the RL method. Moreover, looking at Fig. 2(b)–(d), both the MADRL and DRL methods enable defenders to receive more utility than the NE suggests, while defenders using the RL method obtain almost the same utility as, and sometimes even less utility than, the NE suggests. This result shows that defenders using both the MADRL and DRL methods can beat the attackers, whereas defenders using the RL method cannot.

2) Situation 2: Fig. 3 demonstrates how each method performs with the number of attackers varied from 2 to 5. The number of hosts is set to 4. The average utility of each host is set to 2, where the utility of each host is uniformly drawn from [1, 3]. The number of defenders is set to 4. The average number of resources of each defender and attacker is set to 4, where the number of resources of each defender is uniformly selected from [3, 5].

In Fig. 3(a), when the number of attackers increases, the average utility of the defenders gradually reduces in the three methods. As the number of attackers increases, the number of overall resources of the attackers also increases. Defenders, thus, are difficult to beat attackers regardless of what methods these defenders use. However, by taking advantage of deep reinforcement learning and experience sharing, the MADRL method does create better strategies for defenders than the other two methods.

Fig. 3(b)–(d) shows how the defenders' average utility change for each methods as the game progresses. By comparing these three figures with Fig. 2(b)–(d), we can see that the convergence speed of each method in Situation 2 is faster than in Situation 1. For example, by comparing Figs. 2(b) and 3(b),

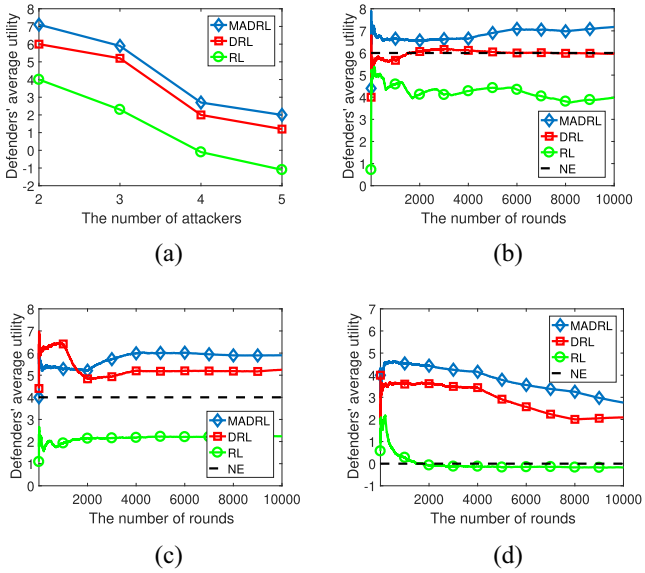

Fig. 3. Performance of the three methods in Situation 2. (a) Defenders' average utility with different number of attackers. (b) Defenders' average utility when attackers = 2. (c) Defenders' average utility when attackers = 3. (d) Defenders' average utility when attackers = 4.

the convergence of the MADRL method starts at 8000 learning rounds in Situation 1 [Fig. 2(b)] while starts at 2000 learning rounds in Situation 2 [Fig. 3(b)]. This can be explained by the fact that in Situation 2, increasing the number of attackers only raises the number of states, while in Situation 1, increasing the number of hosts raises both the number of states and the number of strategies to each defender.

3) Situation 3: Fig. 4 demonstrates how each method performs when the defenders have different amounts of resources. The average number of resources of each defender varies from 3 to 6, where uniform selection range varies from [2, 4] to [5, 7]. The average number of resources of each attacker is set to 4, where the number of resources of each attacker is uniformly selected from [3, 5]. The numbers of defenders and attackers are both set to 4. The number of hosts is set to 4. The average utility of each host is set to 2, where the number of resources of each host is uniformly drawn from [1, 4].

As shown in Fig. 4(a), when the number of resources increases, defenders' average utility steadily increases with each method, while MADRL consistently outperforms the baselines. Since the number of resources of each defender increases, they have more opportunity to beat the attackers. It should also be noted that when each defender has an average of more than four resources, their average utility rises very slowly. This is mainly due to a property of the game, in that when the defenders allocate more resources to a host than the attackers, the defenders take control of the host. This property implies that the disparity in resources allocated to a host does not affect the defenders' utility. Thus, when defenders beat attackers, simply increasing the number of resources available to the defenders does not raise their utility.

By comparing Fig. 4(b)–(d), we find that the defenders' average utility rises as the number of the defenders' resources increases. This can be explained by the fact that increasing the

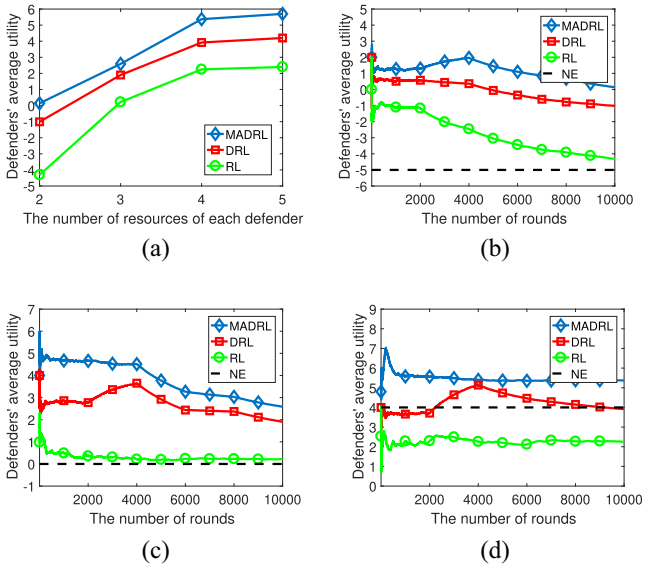

Fig. 4. Performance of the three methods in Situation 3. (a) Defenders' average utility with different number of resources. (b) Defenders' average utility when resources = 3. (c) Defenders' average utility when resources = 4. (d) Defenders' average utility when resources = 5.

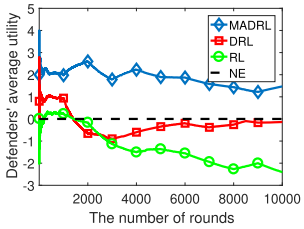

Fig. 5. Performance of the three methods in Situation 4.

number of the defenders' resources also increases the chance that the defenders beat the attackers. Therefore, the defenders' utility rises.

4) Situation 4: Fig. 5 shows how each method performs when the importance values of hosts are dynamically changed as game progresses. The average number of resources of each defender and attacker is set to 4, where the number of resources of each player is uniformly selected from [3, 5]. The numbers of defenders and attackers are both set to 4. The number of hosts is set to 4. The utility of each host is uniformly selected from [1, 3] in every 1000 rounds.

In Fig. 5, due to the dynamism, the three methods have more fluctuations compared to the above three static situations. Particularly, the RL method has been hard to converge in this dynamic situation. In comparison, the MADRL and DRL methods can still achieve decent results due to the utilization of neural networks.

5) Situation 5: Fig. 6(b) and (d) demonstrates how each method performs in the modified game model, i.e., probabilistic winning. For comparison, Fig. 6(a) and (c) demonstrates how each method performs in the regular game model. The average number of resources of each defender and attacker is set to 6, where the number of resources of each defender is uniformly selected from [5, 7]. The numbers of defenders and attackers are both set to 5. The number of hosts is

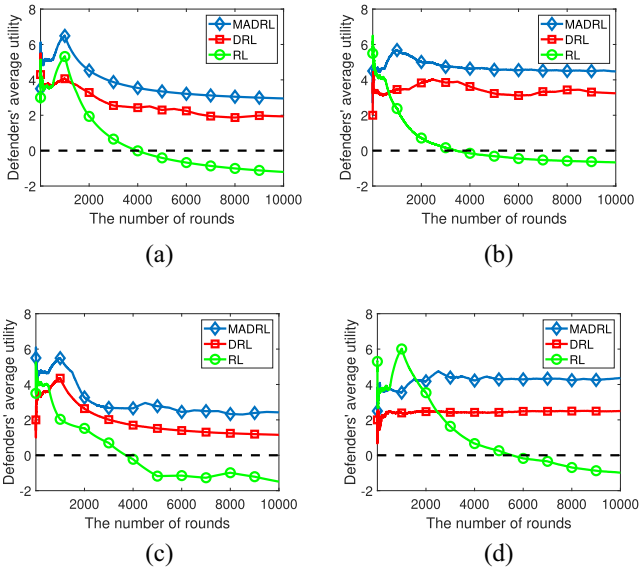

Fig. 6. Performance of the three methods in Situation 5. (a) Defenders' average utility in a static environment. (b) Defenders' average utility in a static environment with probabilistic winning. (c) Defenders' average utility in a dynamic environment. (d) Defenders' average utility in a dynamic environment with probabilistic winning.

set to 5. The average utility of each host is set to 2 and is reassigned in every 1000 rounds in dynamic environments [Fig. 6(c) and (d)], where the utility of each host is uniformly selected from [1, 3] in every 1000 rounds.

By comparing Fig. 6(a) with (b) and comparing Fig. 6(c) with (d), in the three methods, defenders receive more utility in the modified game model than in the regular game model in both static and dynamic environments. For example, in Fig. 6(a), the utility received by defenders under MADRL is 3.6, while in Fig. 6(b), the utility under MADRL is 4.2, about 17% increase. According to the attackers' strategy, they prefer to allocate more resources on more important hosts. Thus, in the regular game model, defenders are less likely to beat attackers on more important hosts. Then, if a defender learns that she is hard to win on a host, she will allocate 0 resource on that host. However, in the modified game model, even if an attacker allocates a large number of resources on a host, a defender still has a chance to win on that host by allocating a reasonable amount of resources. Therefore, compared with the regular game model, defenders are more likely win on important hosts in the modified game model. This increases defenders' utility.

B. Summary

According to the experimental figures (Figs. 2–6), it can be seen that in most situations, there is a big difference between the learning results of our MADRL method and the NE, represented as the black dashed line in these figures. The NE is an equilibrium for both parties but not an optimal result for either party. As each party aims to maximize its own utility by defeating the other party, our method is designed for this aim for defenders. By using our method, defenders can receive a much higher utility than the utility they can receive at the NE

point. In other words, defenders, using our method, can create appropriate strategies to defeat attackers by taking attackers' previous strategies into consideration.

Due to the adoption of MADRL technique, our MADRL method outperforms DRL and RL in various situations by significantly improving defenders' utility. In all the situations, defenders' utility is above the equilibrium with our MADRL method. In comparison, in some situations, defenders' utility can only achieve but not exceed the equilibrium with the DRL method, while in the RL method, defenders' utility is under the equilibrium. Such results demonstrate that defenders can beat attackers using our MADRL method, even if defenders are in disadvantageous situations, such as that defenders have less resources than attackers. Specifically, in average, our method achieves 15% and 40% more utility than DRL and RL, respectively. Moreover, our method converges 8% and 15% faster than DRL and RL, respectively, in all the experimental situations.

In the experiments, all the three methods converge before 10 000 rounds, i.e., their results remain almost unchanged. In some figures, the results seem un-converged, e.g., the MADRL in Fig. 3(d). This is because these results start to converge in the last few hundred rounds. Since the experimental figures are highly shrunk, e.g., 1 cm in the x-axis standing for 2000 rounds, the convergence is not obvious for human perception if it happens in the last few hundred rounds.

We have also evaluated the three methods in the situation with 20 hosts, 20 defenders and 20 attackers. Their performance in this situation has a similar trend to the situation with five hosts, five defenders, and five attackers. However, the three methods need a longer learning process to converge in the situation with more hosts than in the situation with less hosts. This is because, as discussed in the complexity analysis (Section V-C), more hosts incur more strategies and, thus, a larger action space, for each defender. A larger action space, accordingly, implies a larger exploration space and, hence, a longer learning process. Due to the page limitation, the experimental results are not graphically presented.

VII. Conclusion and Future Work

In this article, we designed and investigated a multidefender, multiattacker game model for APT attacks, and proposed an MADRL method for defenders to create efficient strategies. This article was the first which studies multidefender, multiattacker cyber security game and takes the advantages of MADRL to yield efficient defense strategies. Compared with benchmark methods, our MADRL method brings defenders more utility in various situations.

In the future, we will improve our method by introducing knowledge transfer between defenders. In this article, defenders can share experience but they cannot transfer the weights of DQN between each other. These weights also contain defenders' knowledge. Hence, an efficient knowledge transfer method is worth to be developed.

References

[1] M. Wang, T. Zhu, T. Zhang, J. Zhang, S. Yu, and W. Zhou, "Security and privacy in 6G networks: New areas and new challenges," *Digit. Commun. Netw.*, vol. 6, no. 3, pp. 281–291, 2020.

[2] D. Zhang and G. Feng, "A new switched system approach to leader–follower consensus of heterogeneous linear multiagent systems with DoS attack," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 2, pp. 1258–1266, Feb. 2021.

[3] L. Zhang, T. Zhu, P. Xiong, W. Zhou, and P. S. Yu, "More than privacy: Adopting differential privacy in game-theoretic mechanism design," *ACM Comput. Surveys*, vol. 54, no. 7, pp. 1–37, 2021.

[4] C. Tankard, "Advanced persistent threats and how to monitor and deter them," *Netw. Security*, vol. 8, no. 8, pp. 16–19, 2011.

[5] D. Ye, T. Zhu, S. Shen, and W. Zhou, "A differentially private game theoretic approach for deceiving cyber adversaries," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 569–584, 2021.

[6] L. Yang, P. Li, Y. Zhang, X. Yang, Y. Xiang, and W. Zhou, "Effective repair strategy against advanced persistent threat: A differential game approach," *IEEE Trans. Inf. Forensics Security*, vol. 14, pp. 1713–1728, 2019.

[7] R. Brewer, "Advanced persistent threats: Minimizing the damage," *Netw. Security*, vol. 2014, no. 4, pp. 5–9, 2014.

[8] L. Xiao, D. Xu, C. Xie, N. B. Mandayam, and H. V. Poor, "Cloud storage defense against advanced persistent threats: A prospect theoretic study," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 3, pp. 534–544, Mar. 2017.

[9] L. Xiao, D. Xu, N. B. Mandayam, and H. V. Poor, "Attacker-centric view of a detection game against advanced persistent threats," *IEEE Trans. Mobile Comput.*, vol. 17, no. 11, pp. 2512–2523, Nov. 2018.

[10] T. Alpcan and T. Basar, *Network Security: A Decision and Game Theoretic Approach*. Cambridge, U.K.: Cambridge Univ. Press, 2010.

[11] M. van Dijk, A. Juels, A. Oprea, and R. L. Rivest, "FlipIt: The game of 'stealthy takeover,"' *J. Cryptol.*, vol. 26, no. 4, pp. 655–713, 2013.

[12] E. A. Panaousis, A. Fielder, P. Malacaria, C. Hankin, and F. Smeraldi, "Cybersecurity games and investments: A decision support approach," in *Proc. GameSec*, 2014, pp. 266–286.

[13] S. Rass, S. Konig, and E. Panaousis, "Cut-the-rope: A game of stealthy intrusion," in *Proc. GameSec*, 2019, pp. 404–416.

[14] F. Moisan and C. Gonzalez, "Security under uncertainty: Adaptive attackers are more challenging to human defenders than random attackers," *Front. Psychol.*, vol. 8, pp. 1–10, Jun. 2017.

[15] J. Gan, E. Elkind, and M. Wooldridge, "Stackelberg security games with multiple uncoordinated defenders," in *Proc. AAMAS*, 2018, pp. 703–711.

[16] H. Fang, L. Xu, and X. Wang, "Coordinated multiple-relays based physical-layer security improvement: A single-leader multiple-followers Stackelberg game scheme," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 197–209, 2018.

[17] L. Huang and Q. Zhu, "Analysis and computation of adaptive defense strategies against advanced persistent threats for cyber-physical systems," in *Proc. GameSec*, 2018, pp. 205–226.

[18] S. Sengupta, A. Chowdhary, D. Huang, and S. Kambhampati, "General sum Markov games for strategic detection of advanced persistent threats using moving target defense in cloud networks," in *Proc. GameSec*, 2019, pp. 492–512.

[19] R. Zhang and Q. Zhu, "FlipIn: A game-theoretic cyber insurance framework for incentive-compatible cyber risk management of Internet of Things," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2026–2041, 2020.

[20] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1851–1877, 2nd Quart., 2019.

[21] M. Min, L. Xiao, C. Xie, M. Hajimirsadeghi, and N. B. Mandayam, "Defense against advanced persistent threats in dynamic cloud storage: A colonel blotto game approach," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4250–4261, Dec. 2018.

[22] N. Kamra, U. Gupta, K. Wang, F. Fang, Y. Liu, and M. Tambe, "DeepFP for finding nash equilibrium in continuous action spaces," in *Proc. GameSec*, 2019, pp. 238–258.

[23] N. Dunstatter, A. Tahsini, M. Guirguis, and J. Tesic, "Solving cyber alert allocation Markov games with deep reinforcement learning," in *Proc. GameSec*, 2019, pp. 164–183.

[24] T. Eghtesad, Y. Vorobeychik, and A. Laszka, "Adversarial deep reinforcement learning based adaptive moving target defense," in *Proc. GameSec*, 2020, pp. 58–79.

[25] Y. Lv and X. Ren, "Approximate nash solutions for multiplayer mixed-zero-sum game with reinforcement learning," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 12, pp. 2739–2750, Dec. 2019.
[26] M. Zhang, F. Eliassen, A. Taherkordi, H. Jacobsen, H. Chung, and Y. Zhang, "Demand–response games for peer-to-peer energy trading with the hyperledger blockchain," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 1, pp. 19–31, Jan. 2022.
[27] J. Zhang, J. Sun, and C. Zhang, "Stochastic game in linear quadratic Gaussian control for wireless networked control systems under DoS attacks," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 2, pp. 902–910, Feb. 2022.
[28] G. Bonanno, *Game Theory*, 2nd ed. Scotts Valley, CA, USA: CreateSpace Independent Publ. Platform, 2018.
[29] E. Boix-Adsera, B. L. Edelman, and S. Jayanti, "The multiplayer colonel blotto game," *Games Econ. Behav.*, vol. 129, pp. 15–31, Sep. 2021.
[30] A. Sinha, F. Fang, B. An, C. Kiekintveld, and M. Tambe, "Stackelberg security games: Looking beyond a decade of success," in *Proc. IJCAI*, 2018, pp. 5494–5501.
[31] B. Roberson, "The colonel blotto game," *Econ. Theory*, vol. 29, no. 1, pp. 1–24, 2006.
[32] A. Gilpin and T. Sandholm, "Solving two-person zero-sum repeated games of incomplete information," in *Proc. AAMAS*, 2008, pp. 903–910.
[33] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[34] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[35] D. Ye, T. Zhu, W. Zhou, and P. S. Yu, "Differentially private malicious agent avoidance in multiagent advising learning," *IEEE Trans. Cybern.*, vol. 50, no. 10, pp. 4214–4227, Oct. 2020.
[36] D. Ye, T. Zhu, Z. Cheng, W. Zhou, and P. S. Yu, "Differential advising in Multiagent reinforcement learning," *IEEE Trans. Cybern.*, vol. 52, no. 6, pp. 5508–5521, Jun. 2022, doi: 10.1109/TCYB.2020.3034424.
[37] D. Ye, T. Zhu, C. Zhu, W. Zhou, and P. S. Yu, "Model-based self-advising for multi-agent learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Feb. 14, 2022, doi: 10.1109/TNNLS.2022.3147221.
[38] H. Yin and J. Pan, "Knowledge transfer for deep reinforcement learning with hierarchical experience replay," in *Proc. AAAI*, 2017, pp. 1640–1646.
[39] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. ICLR*, 2016, pp. 1–21.
[40] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *Proc. AAAI*, 2018, pp. 3302–3309.
[41] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. NIPS*, 2016, pp. 1–9.
[42] J. Foerster et al., "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proc. ICML*, 2017, pp. 1146–1155.
[43] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep Q-learning," 2020, *arXiv:1901.00137*.
[44] Z. Xu, X. Cao, and X. Gao, "Convergence analysis of gradient descent for eigenvector computation," in *Proc. IJCAI*, 2018, pp. 2933–2939.
[45] S. Feng, Z. Xiong, D. Niyato, and P. Wang, "Dynamic resource management to defend against advanced persistent threats in fog computing: A game theoretic approach," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 995–1007, Jul.–Sep. 2021, doi: 10.1109/TCC.2019.2896632.

Tianqing Zhu received the B.Eng. degree in applied chemistry and M.Eng. degree in detection technology and automation from Wuhan University, Wuhan, China, in 2000 and 2004, respectively, and the Ph.D. degree in computer science from Deakin University, Burwood, VIC, Australia, in 2014.

She is currently a Professor with the School of Computer Science, China University of Geosciences, Wuhan. Her research interests include privacy preserving, data mining, and network security.

Dayong Ye received the M.Sc. and Ph.D. degrees in computer science from the University of Wollongong, Wollongong, NSW, Australia, in 2009 and 2013, respectively.

He is currently a Research Fellow of Cyber-Security with the University of Technology Sydney, Ultimo, NSW, Australia. His research interests focus on differential privacy, privacy preserving, and multiagent systems.

Zishuo Cheng received the B.I.T. degree in information technology and M.Comp. degree in computer science from The Australian National University, Canberra, ACT, Australia, in 2013 and 2015, respectively. He is currently pursuing the Ph.D. degree with the School of Computer Science, University of Technology Sydney, Ultimo, NSW, Australia.

His research interests include privacy preserving, multiagent system, and machine learning.

Wanlei Zhou received the B.Eng. and M.Eng. degrees in computer science and engineering from the Harbin Institute of Technology, Harbin, China, in 1982 and 1984, respectively, the Ph.D. degree in computer science and engineering from The Australian National University, Canberra, ACT, Australia, in 1991, and the D.Sc. degree in computer science from Deakin University, Burwood, VIC, Australia, in 2002.

He was an Alfred Deakin Professor and the Chair of Information Technology with Deakin University, and was the Head of the School of Computer Science, University of Technology Sydney, Ultimo, NSW, Australia. He is currently a Vice-Rector with the City University of Macau, Macau, China. He has published more than 300 papers in refereed international journals and refereed international conferences proceedings. His research interests include distributed systems, network security, and privacy preserving.

Philip S. Yu (Fellow, IEEE) received the B.S. degree in electrical engineering (E.E.) from National Taiwan University, Taipei, Taiwan, in 1972, the M.S. and Ph.D. degrees in E.E. from Stanford University, Stanford, CA, USA, in 1976 and 1978, respectively, and the M.B.A. degree in business administration from New York University, New York, NY, USA, in 1982.

He is a Distinguished Professor of Computer Science with the University of Illinois at Chicago, Chicago, IL, USA, and also holds the Wexler Chair of Information Technology. He has published more than 1000 papers in refereed journals and conferences. He holds or has applied for more than 300 U.S. patents. His research interests are on big data, including data mining, data stream, database, and privacy.

Dr. Yu is a Fellow of ACM.