Hindawi
International Journal of Intelligent Systems
Volume 2023, Article ID 5560416, 15 pages
https://doi.org/10.1155/2023/5560416

Research Article

Flipit Game Deception Strategy Selection Method Based on Deep Reinforcement Learning

Weizhen He,[1,2] **Jinglei Tan**,[1] **Yunfei Guo**,[1] **Ke Shang**,[1,2] **and Guanhua Kong**[1,2]

[1]*PLA Information Engineering University, Zhengzhou 450001, China*
[2]*Purple Mountain Laboratories, Nanjing 211100, China*

Correspondence should be addressed to Jinglei Tan; nxutjl@126.com

Received 17 March 2023; Revised 9 July 2023; Accepted 22 July 2023; Published 7 September 2023

Academic Editor: Paolo Gastaldo

The existing cyber deception decision-making model based on game theory primarily focuses on the selection of spatial strategies, which ignores the optimal defense timing and can affect the execution of a defense strategy. Consequently, this paper presents a method for selecting deception strategies based on a multi-stage Flipit game. Firstly, based on the analysis of cyber deception attack and defense, we propose a concept of moving deception attack surface and analyze the characteristics of deception attack and defense interaction behaviors based on the Flipit game model. The Flipit game model is then utilized to create a single-stage deception spatial-temporal decision-making model. Additionally, we introduce the discount factor and transition probability based on a single-stage game model and construct a multi-stage cyber deception model. We provide the utility function of the multi-stage game model, and design a Proximal Policy Optimization algorithm based on deep reinforcement learning to compute the defender's optimal spatial-temporal strategies. Finally, we utilize an application example to validate the effectiveness of the model and the advantages of the proposed algorithm in generating the multi-stage cyber deception strategy.

1. Introduction

With the development of technology and tools of cyber attacks, it presents the characteristics of complexity, concealment, and persistence, such as Advanced Persistent Threats [1] (APT). However, information systems with static attributes, such as cloud computing environments, mainly use passive defensive mechanisms (like Access Control [2] and Intrusion Detection [3]), which are insufficient to resist attackers. Therefore, adaptive and sophisticated attackers often have asymmetric advantages of resources (e.g., time, and prior knowledge of the vulnerabilities) over the defender. In order to change this situation, cyber deception [4], as an active defense technology, misleads the attacker's decision-making process by manipulating the attacker's cognition, enabling the attacker to choose the suboptimal attack behavior. So deception can effectively delay and block the continuity of the attack process. Currently, cyber deception has been frequently applied to Enterprise Networks [5–7], Cyber-Physical Systems [8, 9], Cloud Environments [10–13], the Internet of Things [14, 15], and Software Defined Networks [16–18].

Despite the broad prospects of application scenarios for cyber deception techniques, there are still two challenges in implementing optimal deception strategies. On the one hand, when deploying deception assets, the defender needs to consider diverse and heterogeneous deception configurations to maximize the deception effect and disrupt the attacker's perception of the system attack surface. On the other hand, the defender needs to consider the migration cost of deception assets to prevent excessive resource costs caused by too frequent deception resource migrations. Therefore, given the above challenges, to balance the deception effectiveness and deception cost, the defender must consider both the best deception configuration and the best migration period simultaneously. To this end, a deception model must precisely characterize the defender's spatial-temporal decision-making.

However, most deception decision-making models focus on spatial decision-making, such as the spatial deception

decision-making model based on attack graphs [19–21] and the spatial deception decision-making model based on game theory [22–31]. Among them, the spatial deception decision-making model based on attack graph models the attacker's invasion process and path as an attack graph model. When the defender deploys deception assets in the system, they can trap the attacker to invade according to the attack graph after deploying deception resources and then discover and block the attacker's attack process. The deception decision-making model based on the attack graph can be traced back to the general attack graph model proposed by Cohen and Koike [19]. In this literature, the defender guides the attacker to enter the designed path by designing a set of deception resources. After that, various attack graph models have been proposed to model deception, such as multi-layer attack graphs and Bayesian attack graphs. For example, Milani et al. [20] proposed using a state-based Bayesian attack graph to represent the attacker's path. The defender adopts two types of spatial strategies to manipulate the attack graph: deception and protection, which changes the attack path found by the attacker through reconnaissance and makes the attacker unable to find critical assets. Sayari et al. [21] proposed to use a multi-layer attack graph to model the deception. Based on the assumption that the defender takes such deceptive actions as setting up false network services, creating bread crumbs or honey coins, creating false local or domain accounts, and creating bait files or documents, the optimal deceptive resource placement strategy of the defender is obtained. The deception decision-making model based on game theory models the attacker and defender as game players, constructs the strategy space and utility function of the attacker and defender in the background of reasonable assumptions, and then calculates the Nash equilibrium to obtain the optimal deception strategy. Currently, the main game models are divided into dynamic game and Stackelberg game. Carroll and Grosu [22] proposed to use the signal game to model the interaction between attacker and defender, then obtained the optimal deception strategy by calculating perfect Bayesian equilibrium. Huang and Zhu [23] proposed a dynamic Bayesian game with two-sided incomplete information to model the attack process of advanced persistent attacks. The optimal deception strategy was obtained by using perfect Bayesian Nash equilibrium. Ahmadi et al. [24] proposed to use a two-player partially observable stochastic game to model the defender and attacker and used a mixed-integer linear program and the infiltration strategy to calculate the robust deception strategy. Ye et al. [25] proposed a differential privacy dynamic game approach to model the behavior of deceiving defenders and attackers. In the approach, defenders strategically changed the number of systems and obfuscated the configuration of systems through the differential privacy mechanism, and attackers used Bayesian inference to infer the proper configuration of systems. Using this method, the impact of system quantity changes on network security can be effectively addressed while the attackers with different attack capabilities can be effectively defended. Schlenker et al. [26] introduced a novel deception game model. This game model uses a zero-sum Stackelberg game to model the interaction between the cyber deception defender and the rational attacker. The author proved that computing the optimal deception strategy is NP-hard and provided a mixed-integer linear program to compute the optimal deception strategy. Yin et al. [27] proposed to use the Stackelberg game to model the behavior of attacker and defender. The author analyzed the strategies of attacker and defender from pure and mixed strategy and proposed an algorithm based on ORIGAMI to get the optimal defense strategy. Anjum et al. [28] introduced a method of using deception traffic to confuse the network information obtained by attackers through network reconnaissance. In order to obtain the optimal deception traffic placement strategy, the author used a two-person non-zero-sum Stackelberg game to model the actions of the attacker and defender and verified the effectiveness of this method by constructing the Mininet experimental environment. Ngo et al. [29] proposed to model the behavior of both attackers and defenders using a Stackelberg game on a large active directory attack graph, in which the defender employs a set of honeypots to prevent the attacker from discovering high-value targets.

The above researches on the deception decision-making problem only consider spatial decision-making, simplifying and ignoring the impact of temporal decision-making influence on the deception defense process. Therefore, with the development of AI, some deception decision-making research based on temporal and spatial have emerged. These methods use AI technologies to perceive the occurrence of attacks and then change the deployment of deception strategies. For example, Dowling et al. [32] proposed an adaptive honeypot deployment strategy based on SARSA. Wang et al. [33] proposed a dynamic deployment strategy for intelligent honeypots based on Q-learning. Furthermore, Abay et al. [34] proposed a honey-data generation method based on deep learning.

All of the above three models can guide the selection of deception strategies. However, the deception decision-making method based on the attack graph does not consider the interaction of attacker and defender when modeling the attackers' behavior. Furthermore, this method only considers the spatial decision-making, which is inconsistent with the actual attack and defense interaction behavior. Although the deception decision-making method based on machine learning considers the optimal time, it needs a large number of data sets, and the availability of data sets will directly affect the detection effectiveness. The deception decision-making method based on the game theory can compensate for the shortcomings of the above two methods. However, at present, this method only considers the optimal strategy based on spatial decision-making, which oversimplifies the strategy space of the defender. In addition, most deception decision-making methods based on the game model only consider the interaction behavior between attacker and defender in a single stage and do not consider the effect of changes in the attacker's behavior on the defender's strategy.

Based on the above shortcomings, we construct a multi-stage deception game model based on the Flipit game,

considering both temporal and spatial decision-making. The Flipit game model provides continuous strategic space for both attacker and defender, which can well describe the timing characteristics of the migration of deception assets. We construct the utility function by introducing the discount factor and the stage transition probability. Based on analyzing the multi-stage network deception model, the optimal deception strategy selection algorithm based on deep reinforcement learning is designed. The effectiveness of this model is verified by simulation experiments.

The main contributions of this paper are as follows:

(1) Different from the moving attack surface and deception attack surface, this paper constructs a deception defense model based on the moving deception attack surface, which can accurately represent the deception attack and defense process.

(2) We analyze the impact of spatial-temporal deception on attackers by modeling the deception decision-making as a Flipit game model and introducing the temporal decision-making into the deception model.

(3) By introducing discount factor and stage transition probability, the single-stage Flipit game model is extended to a multi-stage Flipit game model and then we analyze how the defender's deception strategy adapts to the attacker's strategy changing.

(4) The deep reinforcement learning algorithm is proposed to obtain the multi-stage deception defense strategy. The experimental results verified that the multi-stage Flipit game model can effectively describe and characterize the security evolution process of the deception system.

2. Analysis of Attack and Defense

2.1. Moving Deception Attack Surface. The attack surface can effectively present the system resource vulnerability set as an evaluation method to describe the system security status and potential security risks. Therefore, it is often used to model network attack and defense. For example, in literature [35], Mandahata first defined attack surface. The attack surface is a part of the system resources, mainly including methods M, channels C, and data I. Attackers can use the system's channel to connect, call the system's method, send data to system or receive data from the system, and launch the attack. Then, the author introduces the application of attack surface evaluation in Moving Target Defense (MTD) and puts forward the concept of attack surface shuffling. By shuffling the system attack surface, the defender can effectively reduce the time of system resource leakage and the probability of attack success. However, the shuffling system attack surface proposed by Mandahata et al. assumed the system attack surface is time-invariant, which is not in line with the characteristics of (MTD). Therefore, Huang and Ghosh [36] proposed the concept of moving attack surface with time-varying characteristics, which makes it impossible for attackers to determine whether the vulnerability of resources they use can be reached in a fixed period. Later, Albanese et al. [37] and Ma et al. [38] introduced the attack surface into the cyber deception and proposed the concept of virtual attack surface to represent the attacker's view of system resources and the concept of deception attack surface to represent the attack surface observed and perceived by the attacker. Unlike the moving attack surface used in MTD, the attack surface used in deception can change the attack surface by adding deception assets and building deception topology, which can accurately represent the perception view of attackers to system resources. However, the existing concept of attack surface can not represent the scenario of deception assets migration, so based on the concepts of the deception attack surface and the moving attack surface, we propose the concept of the moving deception attack surface for the scenario in this paper.

As shown in Figure 1, unlike the moving attack surface and the deception attack surface, the moving deception attack surface proposed in this paper shuffles the deception attack surface of the system instead of shuffling the real attack surface of the system, which means shuffling the deception assets to build a dynamic multi-dimensional deception topology. This method can increase the space for attackers to explore by building various heterogeneous deception assets in the spatial dimension. Moreover, the information collected by the attacker is expired by shuffling deception assets in the temporal dimension, which increases the uncertainty of the attacker's perception of the vulnerability of the system resources. According to the definition of Lei et al. [39], this paper defines the moving deception attack surface as follows:

Definition 1. Moving Deception Attack Surface (MDAS) is composed of a triple $\langle \mathrm{DASD}, \mathrm{DASV}, \mathrm{T_d} \rangle$, which represents the vulnerability set of network resources that exposed by the system to the attacker in any given shuffling period $\mathrm{T_d}$. The vulnerability set includes the union of real assets and deception assets. Among them, the Deception Attack Surface Dimension (DASD) refers to the type of network resources in the target system, including the type of deception and real resources. And the network resources are usually in the form of network address, port, service, protocol, etc. The Deception Attack Surface Value (DASV) refers to the value of deception resource type. Because the MDAS will change with time, the attack surface of the target system at time t can be expressed as $\mathrm{MDAS}(t) = \prod \mathrm{DASD}_i^t \cdot \mathrm{DASV}_i^t$. $\mathrm{DASD}_i^t = \{\mathrm{dasd}_1^t, \mathrm{dasd}_2^t, \cdots, \mathrm{dasd}_k^t, \cdots\}$ represents the type of network resources exposed by the target system at the time t, $\mathrm{dasd}_i^t = \{\mathrm{M}_i^t, \mathrm{C}_i^t, \mathrm{I}_i^t\}$. $\mathrm{DASV}_i^t = \{\mathrm{dasv}_1^t, \mathrm{dasv}_2^t, \cdots, \mathrm{dasv}_k^t, \cdots\}$ represents the value of network resources in different dimensions at the time t. "1" indicates a deceptive attack surface of this dimension at time t.

In the moving deception attack surface, the defender can shuffle the deception attack surface in two ways. One is shuffling the decoy, and the other is changing the camouflage type. At the same time, the defender can obtain the information of the attacker according to the deception topology.

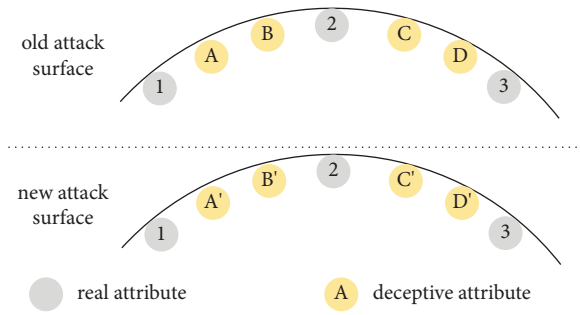

Figure 1: Moving deception attack surface.

2.1.1. Shuffling a Decoy. As for cyber deception, the defender usually deploys honeypots, honeynets, and other deception resources to attract the attacker to deception assets and then delay the attacker's successful attack time. In order to prevent attackers from discovering deception assets, MDAS increases the diversity of system resource vulnerability by shuffling the type of honeypot (such as the transformation of high-interaction honeypot and low-interaction honeypot), the services and vulnerabilities deployed in the honeypot (such as the transformation of Nginx service and Apache service), and the number of honey files deployed in the honeynets.

2.1.2. Changing Camouflage Type. Disguising honest nodes is to confuse attackers through active response, such as allowing the system to simulate different versions of services to respond to attackers or opening all services to respond to attackers' requests. Therefore, the moving deception attack surface can enhance the system's obfuscation by changing the active response service type, making the attacker unable to distinguish between the deception assets and the tangible assets.

2.2. Analysis of Attack and Defense Behavior. In order to evaluate the effectiveness of cyber deception, this paper uses the Flipit game model, as shown in Figure 2, to evaluate the effectiveness of moving deception attack surface. In the Flipit game model, the policies of defender are divided into spatial decision-making and temporal decision-making. The temporal decision-making is referred to the interval at which the defender chooses to control the target system. As shown in Figure 2, the temporal decision-making of defenders can be represented as $\{X_i\}_{i\geq 0}$. The spatial decision-making refers to defenders choosing different deception assets during the interval in which the defender controls the target system. According to the attack and defense strategy defined in the Flipit game model, the assumptions of the attacker and defender are as follows:

(1) Assume that the time interval $\{Y_j\}_{j\geq 0}$ of the attacker to control the target system follows an exponential distribution with parameter λ. The probability distribution of the time interval between attackers getting control of the target system is $p(t) = \lambda e^{-\lambda t}(t \geq 0)$. Within the control time interval $\{Y_j\}_{j\geq 0}$, attackers can detect the attack surface exposed by the target system, and then find resource vulnerability of system to launch further attack.

(2) Because the deployed deception topology can capture part of the information of the attacker, this paper assumes that the defender is the Last Move (LM) defender of the Flipit game. As shown in Figure 1, the defender can see the time interval T from the last time the attacker controlled the target system to the present, and the time LM_0 that he controlled the target system this time. Therefore, LM defenders have certain information advantages over attackers.

(3) Assume that the time interval $\{X_i\}_{i\geq 0}$ for the defender to control the target system is periodic. The defender selects the periodic strategy δ as his own strategy. The deception attack surface can be shuffled within the time interval δ to build a dynamic multidimensional deceiving topology and enhance the deceptive ability of the target system. Different strategies δ of defender will affect its utility. When the strategy δ is larger, the defender can control the target system longer. So, defender can spend a longer time deploying more complex deception assets. So that the smaller the probability of the attacker discovering deception assets is and the greater the utility of the defender is. However, it also results in an increased cost of deploying deception assets at the same time. Therefore, it is necessary to select the optimal strategy that increases the utility of the defender while reducing the cost of the deception assets with high deployment complexity.

3. Multi-Stage Cyber Deception Game Model

In the network attack and defense scenario, when the attacker cannot obtain the critical asset information of the target system for a long time, attacker will actively change his strategy to maximize his utility. Therefore, the single-stage cyber deception attack and defense model does not conform to the actual attack and defense scenario. It is necessary to consider building a multi-stage cyber deception attack and defense model. So in this section, we first give a single-stage Flipit game model based on the literature [40]. Then, a multi-stage Flipit deception attack and defense game model based on the discount reward is given.

3.1. Single-Stage Flipit Deception Game Model. According to the Flipit game model shown in Figure 2, based on the assumption of Section 2.2, we construct the single-stage deception game model at first.

Definition 2. The Flipit-based Single-stage Attack-Defense Deceptive Game Model (FS-ADD) can be represented by triples $FS - \mathrm{ADD} = (N, A, U)$:

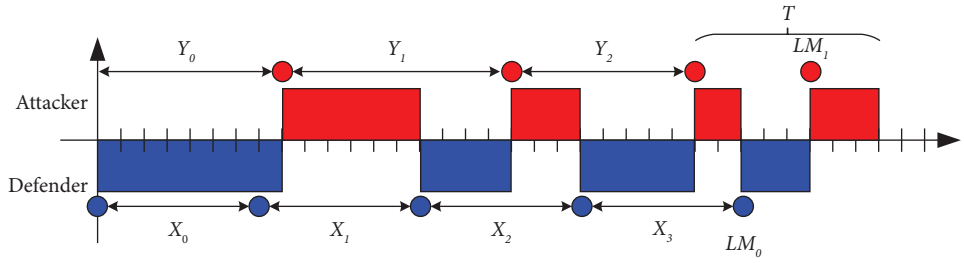

Figure 2: The Flipit game model.

(1) $N = (N_A, N_D)$ refers to the player in the single-stage deception game. We only consider one attacker and one defender, where N_A refers to the attacker, and N_D refers to the defender.

(2) $A = (A_A, A_D)$ refers to the optional strategy set of attacker and defender in the single-stage deception game. A_A represents the strategy set of the attacker. The attackers in this paper are the players who adopt the exponential distribution strategy in the Flipit game. The average time interval between the attacker controlling the target system is $1/\lambda$, so the attackers' strategy is a set of different average time interval $1/\lambda$. A_D represents the strategy set of the defender. We consider that the defender is the LM player who uses periodic strategy δ in Flipit game. Defender's strategy is a set of different time intervals δ.

(3) $U = (U_A, U_D)$ refers to the set of the utility functions of attacker and defender in a single-stage deception game. U_A represents the utility of the attacker. U_D represents the utility of the defender. According to the Flipit game, the utility function of the attacker and defender can be represented as shown in formula:

$$U_A(\lambda, \delta) = 1 - \frac{1 - e^{-\lambda\delta}}{\lambda\delta} - \lambda C_A,$$
$$U_D(\lambda, \delta) = \frac{1 - e^{-\lambda\delta}}{\lambda\delta} - \frac{C_D}{\delta}, \quad (1)$$

where, C_A refers to the cost of the attacker to control the target system, C_D refers to the cost of the defender to control the target system, which means the cost of the defender to change the deception assets.

3.2. Multi-Stage Flipit Deception Game Model. According to the single-stage Flipit game model, the defender is the LM player. So the deception game model exists the information asymmetry between attacker and defender, and there is no dominated strategy for both attacker and defender in the game. This means there is no Nash equilibrium in the deception game. However, the defender can get the strongly dominant strategy. When the system passes through a period, the defender can get the optimal defense strategy and the deception defense system is stable. However, when the attacker cannot obtain the key information of the target system for a long time, the attacker will change his attack strategy. At this time, the strategy of the defender is no longer optimal. Therefore, the relatively stable deception system needs to re-select the best defense strategy of the defender, which means the system moves from one state to the next.

Figure 3 shows a multi-stage Flipit deception game model. In the k stage, we consider that the attacker plays with exponential distribution with λ_k. In order to deceive the attacker and enhance the confusion of the system attack surface, the defender needs to move the deception attack surface periodically to generate a new virtual network topology. However, the selection of the deception strategy will affect the defender's utility. Therefore, the defender must choose the optimal defense strategy to achieve the system's steady state at each stage. However, when the attacker cannot obtain the key asset information of the target system for a period of time, the attacker will change his strategy, making the system shuffle to another stage. The defender needs to re-select the deception strategy to prevent the attacker from obtaining the necessary information from the target system.

Definition 3. The Flipit-based Multi-stage Attack-Defense Deceptive Game Model (FM-ADD) is represented as $FM - \text{ADD} = (N, T, A, S_t, \eta, U)$:

(1) $N = (N_A, N_D)$ refers to the players of the multi-stage deception game. We only consider one attacker and one defender, where N_A refers to the attacker, and N_D refers to the defender.

(2) T refers to the total number of stages in the multi-stage Flipit game, $M(k)$ refers to the current stage, $k = 1, 2, ..., T$.

(3) $A = (A_A^k, A_D^k)$ refers to optional strategy set of attacker and defender in the k-th stage of deception game. A_A^k refers to the optional strategy set of attacker in the k-th stage and A_D^k refers to the optional strategy set of defender in the k-th stage.

(4) $S_0 = (S_0^1, S_0^2, ..., S_0^T)$ refers to the system's initial state at each stage, that is, the security state of the system when the attacker's policy changes.

(5) S refers to the state of the system at each stage, including the state before the system reaches a stable state and the state when each stage reaches stable state $(S_1, S_2, ..., S_T)$.

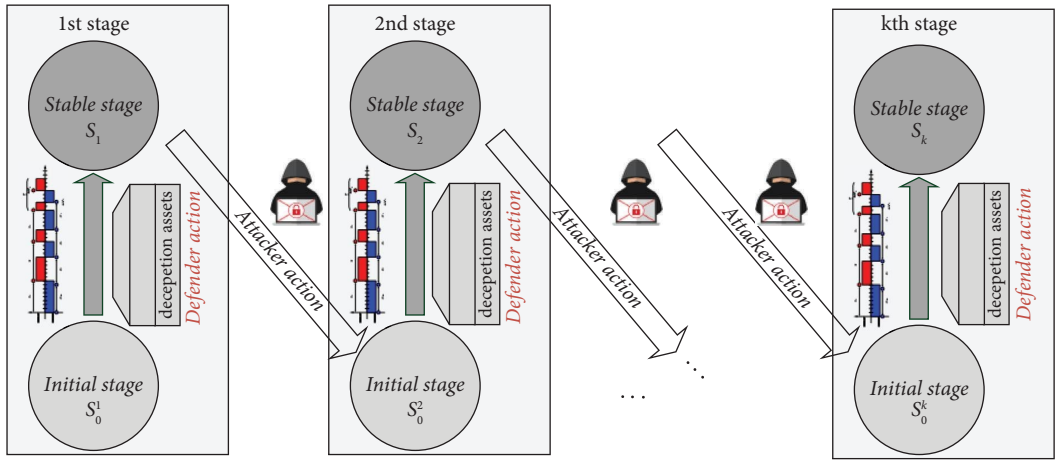

Figure 3: The multi-stage Flipit deception game model.

(6) η refers to the stage transition probability, and $\eta(S_i|S_j)$ refers to the transition probability from steady state S_i to steady state S_j.

(7) $U = (U_A^k, U_D^k)$ refers to the set of the utility functions of attacker and defender in a multi-stage deception game. U_A^k refers to the attacker's utility function in the k-th stage and U_D^k refers to the defender'S utility function in the k-th stage.

In the multi-stage deception game model, the optimal strategy of the defender is not only related to the utility of the current stage, but also to the utility of the future stages. According to the literature [41], we design the objective function to determine the advantages and disadvantages of the strategies of the defender and the attacker by introducing the discount factor and the stage transition probability $\eta(S_i|S_j)$:

$$\begin{cases} R_D^k = U_D^k + \varepsilon \sum_{h\in[0,T]} \eta\left(S_h|S_k\right) R_D^h, \\ R_A^k = U_A^k + \varepsilon \sum_{h\in[0,T]} \eta\left(S_h|S_k\right) R_A^h. \end{cases} \tag{2}$$

4. Cyber Deception Strategy

4.1. Single-Stage Flipit Deception Strategy. According to the single-stage Flipit deception game model, as shown in Figure 2, since the attacker plays with exponential distribution and the defender plays with periodic strategy, the information of the attacker and defender is asymmetric. The defender who deploys the deception strategy can see part of the information of the attacker's strategy, while the attacker cannot obtain the information of the defender. Therefore, the single-stage Flipit deception game model only has the strongly dominated strategy of the defender, and there is no Nash equilibrium between the attacker and defender. According to literature [40], when the attacker and defender are players who follow exponential distribution and LM players who adopt periodic strategy respectively, the defender's strongly dominated strategy can be obtained by calculating the derivative of the defender's utility U_D:

$$\frac{dU_D}{d\delta} = \frac{e^{-\lambda\delta}(1+\lambda\delta) - (1-\lambda C_D)}{\lambda\delta^2}. \tag{3}$$

According to formula (3), when $\lambda < 1/C_D$, the defender has a unique periodic strategy δ_m that maximizes the utility of the defender. The defender's strategy δ_m and the attacker's strategy λ satisfied formula $e^{-\lambda_m\delta}(1+\lambda\delta) = 1 - \lambda C_D$. When $\lambda \geq 1/C_D$, $dU_D/d\delta > 0$. The utility function of the defender is monotonically increasing. So the optimal strategy of the defender is that $\delta \longrightarrow \infty$, which means the defender takes no action to seize control.

4.2. Multi-Stage Flipit Deception Strategy. Based on the single-stage Flipit deception game model, we construct a multi-stage Flipit deception game model by introducing the discount factor ε and the stage transition probability $\eta(S_j|S_i)$. We calculate the optimal defense strategy when the attacker takes different strategies, which means at different stages $M(k)$. The strategy can be obtained as:

$$\begin{aligned} \max R_D^k &= \max\left[U_D^k + \varepsilon \sum_{h\in[0,T]} \eta\left(S_h|S_k\right) R_D^h\right], \\ \mathrm{U}_D^s &= \frac{1-e^{-\lambda\delta}}{\lambda\delta} - \frac{C_D}{\delta}. \end{aligned} \tag{4}$$

When the attacker's strategy meets $\lambda \geq 1/C_D$, the optimal strategy of the defender is $\delta \longrightarrow \infty$. So studying the attacker's strategy that meeting $\lambda \geq 1/C_D$ is meaningless. In this paper, we only considers the optimal deception strategy when the attacker's strategy meets $\lambda < 1/C_D$.

By analyzing formula (4), it can be found that formula (4) is a dynamic programming problem. Because the strategy of the defender is a continuous strategy space, to avoid the dimension disaster problem brought by the traditional algorithm, we propose to use the deep reinforcement learning method to calculate the optimal deception strategy of the multi-stage deception game model. In this section, we first represent the state, action, and reward function of reinforcement learning, then design a Multi-stage Flipit game deception strategy

generation algorithm based on the Proximal Policy Optimization (PPO) algorithm (MFD-PPO).

4.2.1. Reinforcement Learning Algorithm. In the classical reinforcement learning model, the agent interacts with the environment continuously, then gets the optimal goal gradually through training. This process mainly involves the dynamic change of the environment, the behavior of the agent, and the reward after acting, which are called state, action, and reward, respectively. When the agent selects an action A_t from the strategy space π according to the obtained state S_t at a certain time t, it obtains the reward R_t for taking action A_t under state S_t. Then, the state changes from S_t to S_{t+1}. During the whole training process, the interaction between the agent and the environment can be expressed as trajectory $(S_t, A_t, R_t, S_{t+1}, A_{t+1}, \cdots)$, and the effect of training can be quantified by the accumulated discount reward in the learning process as:

$$E(R_t) = E\left(\sum_{t=0}^{T} \gamma^t r_t\right), \tag{5}$$

where, T indicates the number of iterations, γ indicates the discount factor. Formula (5) indicates that the current reward of the agent is not only related to the current reward but also to the future reward. Based on this, in the multi-stage Flipit deception game model, the states S, actions A, and rewards R are represented as follows:

*(1) State S.*In the multi-stage deception game model, the state is composed of the strategy λ taken by the attacker at all stages and the cost of the defender, expressed by S. At time t, the state of each stage of the system can be expressed by $S_t = \{\psi_1^t, \psi_2^t, ..., \psi_k^t\}$. ψ_k^t represents the state of the k-th stage in the multi-stage game process, which can be represented as $\psi_k^t = (\lambda_k^t, C_k^t)$. λ_k^t refers to the strategy of the attacker at the k-th stage, and C_k^t refers to the cost of defender at the k-th stage.

(2) Action A. In the multi-stage deception game model, the action strategy A_D^k of the defender at each stage is used as the action space A. According to the multi-stage deception game model, the deception strategy of the defender is a different fixed time interval δ, which is represented as:

$$A = [T_{\min}, T_{\max}], \tag{6}$$

$T_{\min}$ and $T_{\max}$ represent the minimum and maximum control intervals the defender can choose, respectively. When the defender selects action $A_t \in A_D^k$ at time t, the system will change the configuration of deception nodes according to the selected strategy. If the defender chooses the deception strategy δ, the defender will combine the configuration of k deception node according to the strategy δ. The longer the strategy δ chosen by the defender, the longer it will take to change the deception assets and more complex deception assets can be deployed by k deception nodes.

(3) Reward R. The defender's behavior will affect the security situation of the deception system and obtain rewards from the environment. Here, the objective function of formula (7) is used as the reward function.

$$R = \sum_{k \in T} R_D^k = \sum_{k \in T} \left[U_D^k + \varepsilon \sum_{h \in [0,T]} \eta\left(S_h | S_k\right) R_D^h \right]. \tag{7}$$

4.2.2. Algorithm of Multi-Stage Deception Defense Strategy. Based on the analysis in Section 4.2.1, the action space of the agent is continuous. Using the classical reinforcement learning algorithm, such as Q-learning algorithm, to calculate the deception strategy will lead to the infinite size of the Q table. These reinforcement learning algorithms cannot adapt to the dynamic programming problem of high-dimensional state space and action space. Therefore, in this paper, we design a Multi-Stage Flipit Deception Strategy solution algorithm based on the Proximal Policy Optimization (PPO) algorithm (MFD-PPO). PPO algorithm is a deep reinforcement learning algorithm based on policy gradient. By combining reinforcement learning with neural networks, PPO has apparent advantages in solving the optimization problem of high-dimensional state space and action space.

Figure 4 shows Multi-stage Flipit Deception Game Strategy framework based on the PPO algorithm (MFD-PPO). PPO algorithm is based on the Actor-Critic deep reinforcement learning algorithm, mainly including Actor and Critic neural networks. And the Actor-network includes the new Actor-network and old Actor-network. In the training process, PPO first inputs the state s of the environment into the Actor-network to obtain the expectation and variance required to construct the action space subject to the normal distribution. Then, PPO inputs the random sampling action a into the environment according to the normal distribution and obtains the reward r and the next state $s^{'}$. Further, store (s, a, r) in the experience pool and input state $s^{'}$ into the Actor-network. PPO obtains trajectory $[(s, a, r), \cdots]$ after repeating the above steps. Actor network does not update the network parameters at this time. After the number of state-reward pairs in the experience pool reaches a certain number, PPO inputs the last sampling action to the environment and outputs the state $s^{'}$. Then, PPO inputs the state $s^{'}$ and the state s stored in the experience pool into the Critic-network. According to state $s^{'}$, the Critic network outputs state value $V(s_T)$. The discount reward R_t is represented as formula:

$$R_t = r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T), \tag{8}$$

T is the random sampling for the last step. At the same time, all state values $V(s_t)$ can be obtained according to the state set in the experience pool. Then, the estimate of the advantage function can be obtained as:

$$\hat{A} = r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T) - V(s_t). \tag{9}$$

The loss function of the Critic network can be obtained according to the advantage function shown in equation (9). The system updates parameter ϕ of Critic network according to equation.

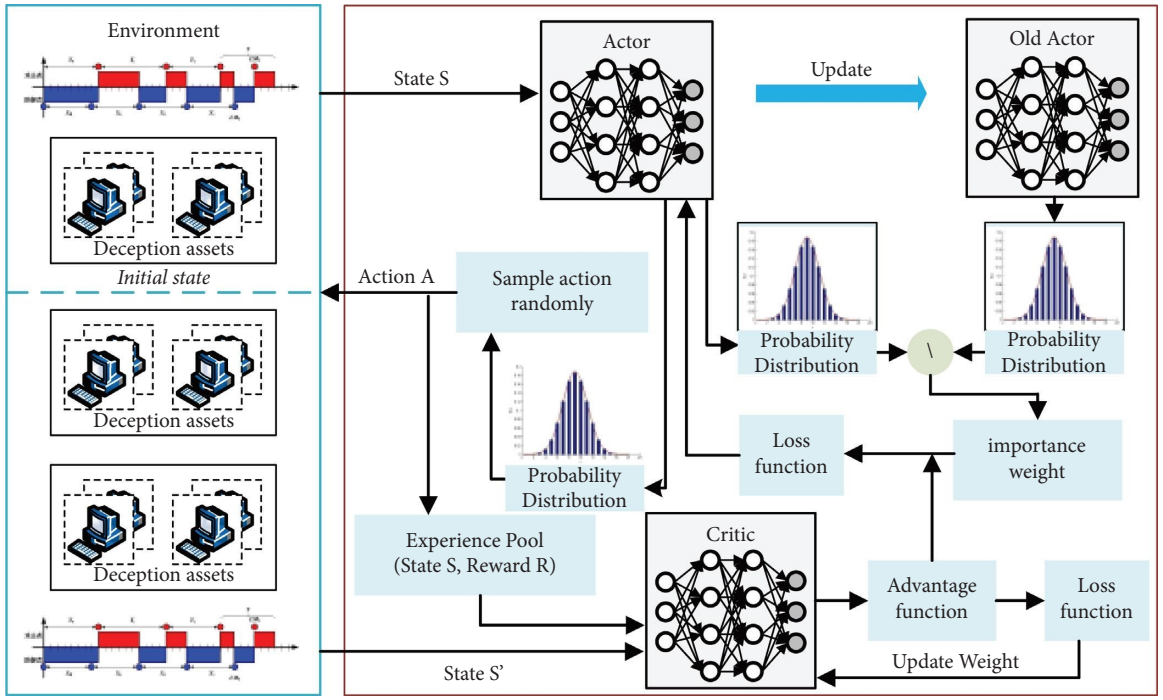

Figure 4: The MFD-PPO algorithm architecture.

$$L(\phi) = E\left[\left(\sum_{i=t}^{T} \gamma^{i-t} r_i - V_\phi(s_t)\right)^2\right]. \tag{10}$$

For the parameter update of the actor-network, the PPO algorithm first collects the combination of state s and inputs them to the new actor-network and the old actor-network. The two networks can output the normal distribution as Normal_1 and Normal_2 respectively. Then, MFD-PPO inputs the stored action set into the normal distribution Normal_1 and Normal_2 to obtain the corresponding probability $\pi_{new}(a_t|s_t)$ and $\pi_{old}(a_t|s_t)$. $\pi_{new}(a_t|s_t)$ divides $\pi_{old}(a_t|s_t)$ to get the importance weight $\pi_\theta(a_t|s_t)/\pi_{\theta_{old}}(a_t|s_t)$. Then, we can get the loss function of actor-network as:

$$L(\theta) = E\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t\right). \tag{11}$$

However, in the training process, if the policy update step is too long, it will affect the optimal value and if the policy update step is too short, it will slow down the convergence speed. Therefore, the PPO algorithm introduces hyper-parameters to clip the policy update step. The loss function of the actor-network is optimized as follows:

$$L(\theta) = E\left[\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\widehat{A}_{\theta'}(s_t|a_t),\right.\right.$$
$$\left.\left.\cdot clip\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon\right)\widehat{A}_{\theta'}(s_t, a_t)\right]. \tag{12}$$

Among them, the clip function can prevent Actor network θ from updating too fast. PPO updates the new Actor network according to formula (12) and repeats the above steps until the iteration is over, then updates the old actor-network. Based on the PPO algorithm framework shown in Figure 4, this paper designs a Multi-stage Flipit Deception Game Strategy Selection Algorithm based on PPO (MFD-PPO). The detailed algorithm is shown in Algorithm 1.

According to the analysis and description of the above algorithm, the MFD-PPO algorithm has strong scalability and efficiency. As for algorithm's scalability, it is known that the MFD-PPO extends from the PPO based on the description of Section 4.2.2. According to the literature [42], the PPO can be applied to scenarios with high-dimensional state and action spaces, so the MFD-PPO can also deal with high-dimensional state and action spaces. To verify the scalability of the MFD-PPO, we set that the action spaces of the defender are $A = [T_{\min}, T_{\max}]$ and the state spaces consist of eight phases of attack policies and costs in Section 4.2.1. So the experimental results can effectively prove that the MFD-PPO can deal with high-dimensional and continuous state and action spaces. Besides, when applying the algorithm to other scenarios, the users only need to reset the state spaces and action spaces, and the algorithm can easily get the results.

As for algorithm's efficiency, we demonstrate that the MFD-PPO can correctly obtain the optimal defense strategy through the experiments in Sections 5.3.1 and 5.3.2. Besides, the MFD-PPO only needs to be trained once. After the training completes, we can deploy the trained model to the target system and use the trained model to easily obtain the optimal deception strategy. Therefore, the computational resource overhead is mainly spent in the training process, and the resources spent are limited.

Input: The attacker's state s_t in cloud environment
Output: Network update parameters θ、 ϕ, Deception strategy δ_t

(1) Set parameters γ, ε
(2) Initialize Actor network parameters as θ_0, Initialize Critic network parameters as ϕ_0
(3) **for** iteration = 1, 2, 3, . . ., **do**
(4) Experience Pool $D \longleftarrow \varnothing$
(5) **for** $k = 1, 2, \ldots$**do**
(6) Randomly initialize state s_0
(7) Initialize state-action trajectory $\tau \longleftarrow \varnothing$
(8) **for** $t = 1, 2, \ldots, T$ **do**
(9) Get current state s_t
//Obtain the strategy taken by the current attacker, i.e. select the exponential distribution strategy λ
(10) Select action a_t from the old Actor network
//Select the interval δ of the defender's control target system
(11) Calculate the reward r_t of multi-stage deception strategy
//Calculate the reward according to formula (5)
(12) Store state-action trajectory $\tau \longleftarrow \tau \cup (s_t, a_t, r_t)$
(13) **end for**
(14) $D \longleftarrow D \cup \tau$
(15) **end for**
(16) Update Actor network policy parameter θ
$L(\theta) = E[\min(\pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)\widehat{A}_{\theta'}(s_t|a_t), \text{clip}(\pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t), 1-\varepsilon, 1+\varepsilon)\widehat{A}_{\theta'}(s_t, a_t)]$
(17) Update Critic network policy parameter ϕ
$L(\phi) = E[(\sum_{i=t}^{T}\gamma^{i-t}r_i - V_\phi(s_t))^2]$
(18) Update the old network parameter: $\theta' \longleftarrow \theta$
(19) **end for**

Algorithm 1:Multi-stage flipit game deception strategy selection algorithm based on proximal policy optimization.

5. Simulation Experiment and Analysis

The rapid development of cloud computing provides users convenient ways to access computing resources. However, the architecture of cloud computing also provides many attack surfaces to attackers, enabling attackers to gain control of the cloud in various ways, then, launching the further attack on the cloud computing environment. Therefore, it is urgent to study the attack and defense model of the cloud environment. In this section, we take the cloud computing environment as an example to analyze the proposed multi-stage deception model based on Flipit game and calculate the multi-stage deception strategy.

5.1. Experimental Environment. In order to build an actual attack and defense interaction scenario in the cloud environment, this paper uses Kubernetes to build an experimental environment. As an automated container management tool, Kubernetes can make the deploying, scheduling, and deleting of applications more convenient. As shown in Figure 5, we build a Kubernetes cluster to manage application services. The cluster mainly includes a master node (16-core processor, 16 G RAM) and three nodes (16-core processor, 16 G RAM). The master node manages the creation and deletion of PODs, and the nodes are used to run PODs. Applications are deployed in PODs that mainly include Web server, FTP server, database server, LADP server, and honeypot server. The clusters of above application servers are the main targets of deception defense. The main defense method is deploying honeypot clusters to prevent attackers from discovering critical assets in the cloud computing environment.

Based on the above experimental environment and reference [43, 44], this paper designs a multi-stage deception game model based on Flipit game which includes eight stages, as shown in Figure 6.

Based on the literature [41, 45], the transition probability between the stages can be obtained based on prior knowledge. Therefore, the transition probability of a multi-stage deception game based on Flipit game is shown in Table 1. Although the given transition probability is fixed in this experiment, they are only used as an example and the set of transition probabilities do not affect the stability of the algorithm.

5.2. Parameters Setting. In order to obtain the multi-stage deception defense strategy, we design our experiments using Python 3.6.7, TensorFlow 1.8.0, and Stable Baselines library for implementing the MFD-PPO algorithm. We have simulated our experiments using HUAWEI machine with Intel (R) Core (TM) i7-6700 @ 3.40 GHz CPU and 32 GB RAM. According to the experimental environment built in Section 5.1, to obtain the 8-stage deception strategy, it is necessary to firstly determine the space of attack and defense strategy before getting the multi-stage deception strategy. Therefore, this section simulates and analyzes the relationship between the defender's reward and the defender's strategy, the defender's cost and the attacker's

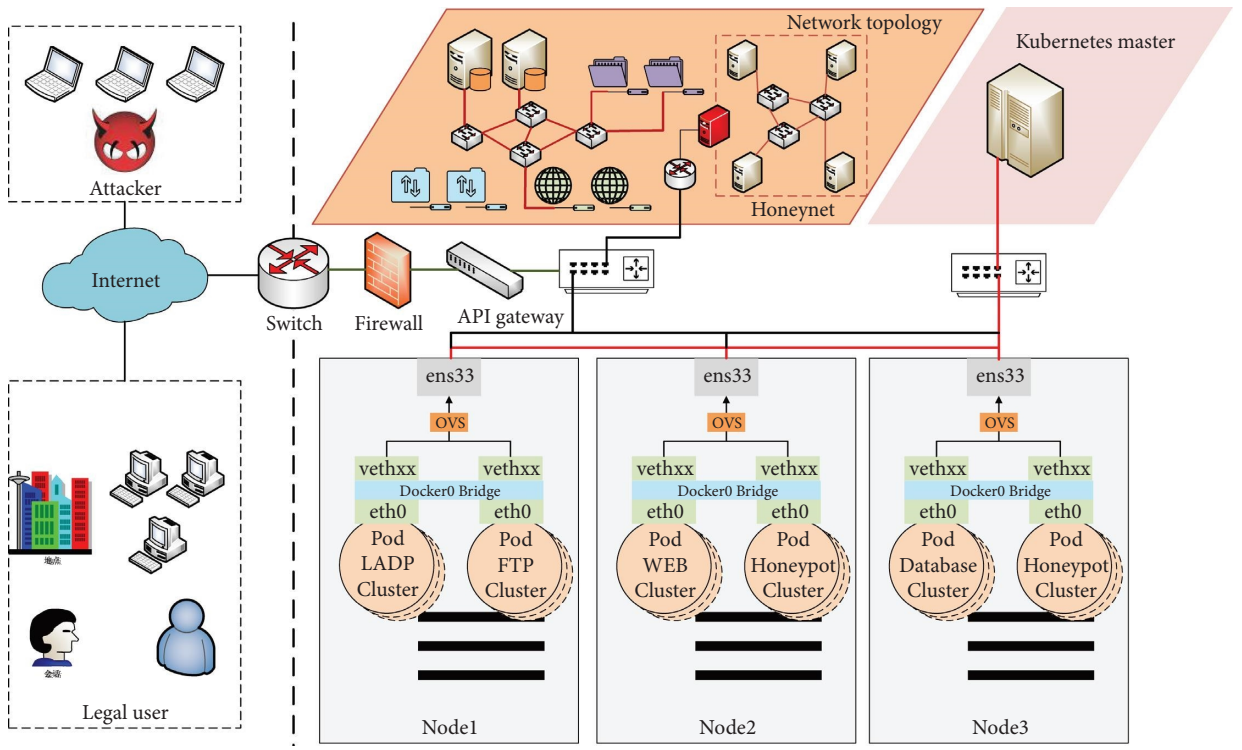

Figure 5: The experimental topology.

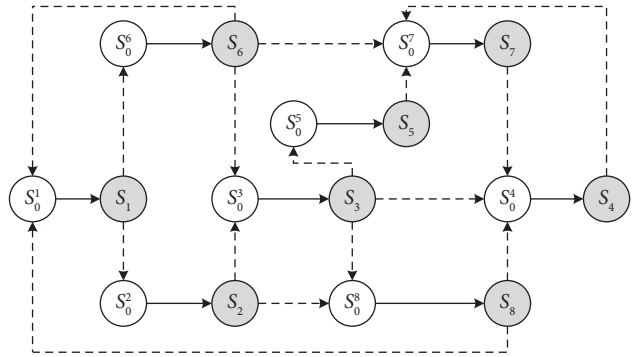

Figure 6: The state transition figure.

Table 1: The transition probability between the stages.

Stage transition	$s_1 \longrightarrow s_0^2$	$s_1 \longrightarrow s_0^6$	$s_2 \longrightarrow s_0^3$	$s_2 \longrightarrow s_0^8$	$s_3 \longrightarrow s_0^4$
Transition probability	$\eta(2\|1) = 0.8$	$\eta(6\|1) = 0.6$	$\eta(3\|2) = 0.7$	$\eta(8\|2) = 0.4$	$\eta(4\|3) = 0.6$
Stage transition	$s_3 \longrightarrow s_0^5$	$s_3 \longrightarrow s_0^8$	$s_4 \longrightarrow s_0^7$	$s_5 \longrightarrow s_0^7$	$s_6 \longrightarrow s_0^1$
Transition probability	$\eta(5\|3) = 0.9$	$\eta(8\|3) = 0.6$	$\eta(7\|4) = 0.8$	$\eta(7\|5) = 0.9$	$\eta(1\|6) = 0.5$
Stage transition	$s_6 \longrightarrow s_0^3$	$s_6 \longrightarrow s_0^7$	$s_7 \longrightarrow s_0^4$	$s_8 \longrightarrow s_0^1$	$s_8 \longrightarrow s_0^4$
Transition probability	$\eta(3\|6) = 0.8$	$\eta(7\|6) = 0.8$	$\eta(4\|7) = 0.6$	$\eta(1\|8) = 0.9$	$\eta(4\|8) = 0.8$

strategy in the single-stage Flipit game model. As shown in Figure 7. when the cost C_D of the defender remains unchanged, the larger the attacker's strategy λ is, the smaller the optimal deception strategy δ is if we want to maximize the reward of the defender. This means that the larger the attacker's strategy λ is, the shorter the average time for the attacker to seize control of cloud is. In order to maximize the reward for the defender, the time interval for the defender to seize control should also be shorter to defend against an attacker with a shorter average shuffle time. At the same time, the smaller the cost of the defender is when the attacker's strategy λ is unchanged, the smaller the interval between the defender seizing control of cloud is when the defender gets the maximum reward.

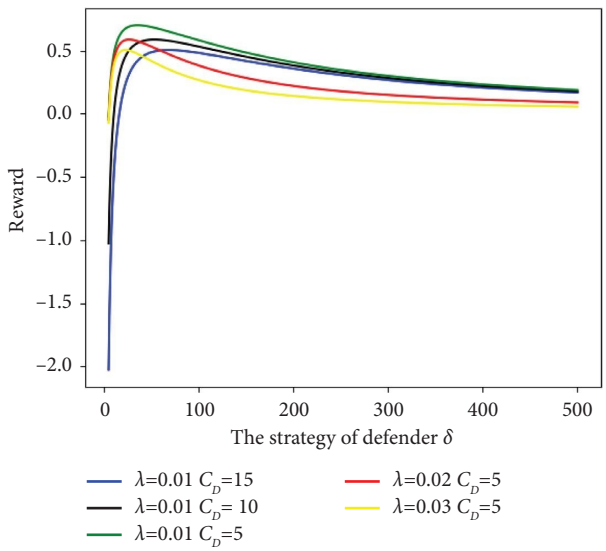

Figure 7: The relationship between the defender's reward and the defender's strategy in a single-stage game.

Just as Section 3.2 described, because the single-stage Flipit game doesn't exist Nash equilibrium and only exists strongly dominated strategy of defender, the multi-stage Flipit game model also doesn't exist Nash equilibrium. To provide robust defense strategies, we set the attacker's strategy to vary within a certain range at each stage during the experiment. This experiment method avoids the influence of fixed attack strategy on the optimal defense strategy and ensures the effectiveness of the defense policy. According to the above simulation results and the analysis in Section 5.1, the attacker's strategy λ must be less than $1/C_D$, and the defender's strategy space must be within 10~50 s. Therefore, when designing the multi-stage deception strategy space, to ensure the defense strategy's accuracy, the strategy space of the defender is 1~100 s, and the strategy space of the attacker is less than $1/C_D$. In addition, to ensure the stability and convergence speed of the MFD-PPO algorithm, the parameters of the MFD-PPO algorithm are designed as shown in Table 2.

In order to compare the advantages of the MFD-PPO algorithm in calculating the multi-stage deception strategy, this paper selects the following algorithms for comparison.

(1) Multi-stage Flipit deception game strategy selection algorithm based on Particle Swarm Optimization (MFD-PSO). PSO algorithm is a typical heuristic algorithm. This algorithm selects the optimal solution by simulating the optimization problem as birds' foraging and flight behavior. The flight space of birds is the solution space and the position of birds in space is one solution to the problem.

(2) Multi-stage Flipit deception game strategy selection algorithm based on Advantage Actor-Critic (MFD-A2C). A2C algorithm is also a deep reinforcement learning algorithm based on Actor-Critic architecture. Multiple parallel threads will be built during the algorithm's training process. Each thread includes an Actor-Critic network and interacts with their own environment to obtain independent experience.

Table 2: The simulation parameters setting.

Simulation parameters	Value
Attacker's strategy λ	[0.01–0.15]
Defender's strategy δ	[1–100]
Defender's cost C_D	5
Hyperparameter ε	0.2
Learning rate γ	2.5×10^4
Batch size	128

5.3. Analysis of Experiment Results

5.3.1. Convergence Analysis. In order to analyze the convergence of the MFD-PPO algorithm and the impact of introducing hyper-parameter ε into the clip function on MFD-PPO algorithm, this simulation carries out 1.2×10^6-step training. Since the hyper-parameters ε in the clip function generally range from 0.1 to 0.3, the hyper-parameters ε selected in the experiment are 0.1, 0.15, 0.2, 0.22, 0.26, and 0.3. The training results are shown in Figure 8, where the horizontal axis represents the number of training steps and the vertical axis represents the discount reward. From the experimental results shown in Figure 8, the MFD-PPO starts to convergence when the training reaches 600000 steps, and the training time is only three hours. Besides, when the hyper-parameter ε is 0.1, 0.15, 0.2, 0.22, 0.26, and 0.3, and the discount reward after convergence is also basically the same. This result can prove the effectiveness of the MFD-PPO. In addition, the experimental results show that the smaller the hyper-parameter ε is, the slower the convergence speed is. This is because in the training process, the smaller the hyper-parameter is, the more cautious the strategy update is, so the slower the convergence speed is. However, the experimental results show that the convergence speed is similar when the hyper-parameter is in the range of 0.2 to 0.3. In order to avoid excessive differences between the new and old policies due to the excessively large hyper-parameter ε, this paper selects the hyper-parameter to be 0.2.

5.3.2. Multi-Stage Deception Strategy Solution. Based on the hyper-parameter ε selected in Section 5.3.1, this section calculates the reward and deception strategy of each stage in the multi-stage Flipit deception model. The experimental results are shown in Figures 9 and 10. Figure 9 shows the defender's reward at each stage. From the reward of the defender at each stage, it can be seen that all of the 8-stages' rewards can converge during the training process. In the case of the reward convergence of the defender, when the attacker takes different attack strategies, the strategy of the defender is shown in Figure 10. As shown in Figures 9 and 10, when the attacker is in the 1st stage and the system reaches a steady state, the defender's reward fluctuates around 1.05, and the defender's strategy δ fluctuates around 20. When the attacker is in the 2nd stage and the system reaches a steady

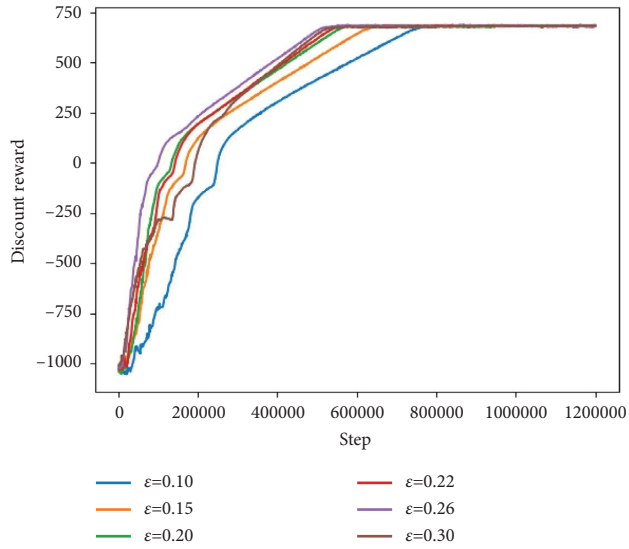

Figure 8: Convergence performance of MFD-PPO under different hyper-parameters ε.

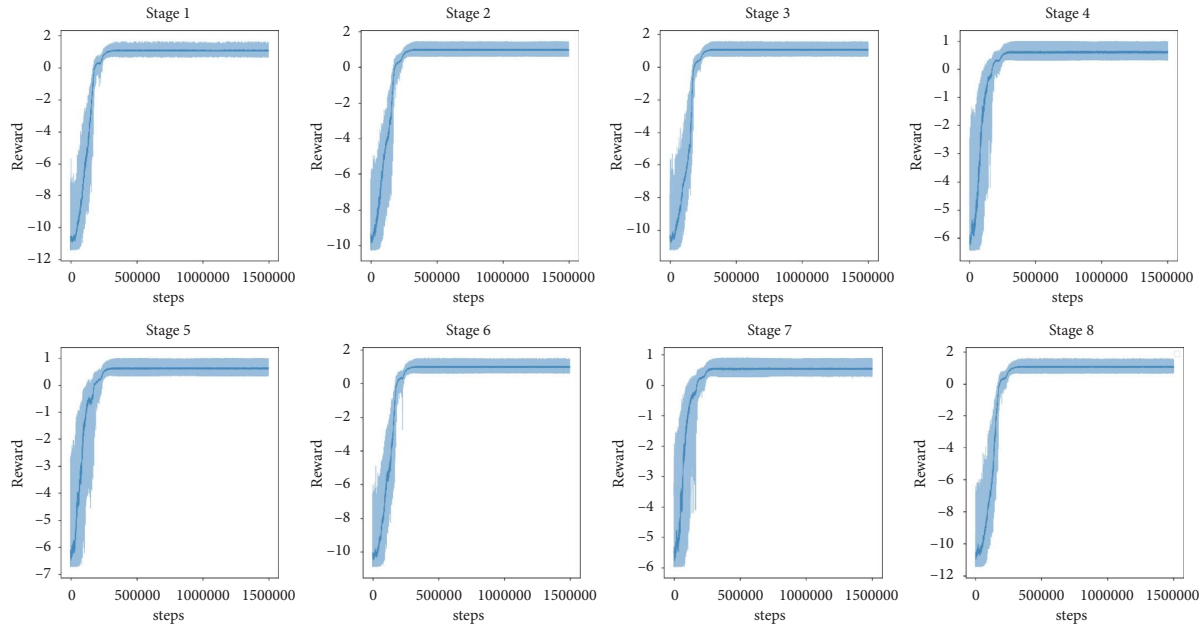

Figure 9: The reward of defender at each stages.

state, the defender's reward is 0.96, and the defender's strategy δ fluctuates around 20. When the attacker is in the 3rd stage and the system reaches a steady state, the defender's reward is 1.02, and the defender's strategy δ fluctuates around 22. When the attacker is in the 4th stage and the system reaches a steady state, the defender's reward is 0.58, and the defender's strategy δ fluctuates around 24. When the attacker is in the 5th stage and the system reaches a steady state, the defender's reward is 0.62, and the defender's strategy δ fluctuates around 20. When the attacker is in the 6th stage and the system reaches a steady state, the defender's reward is 0.99, and the defender's strategy δ fluctuates around 20. When the attacker is in the 7th stage and the system reaches a steady state, the defender's reward is 0.53, and the defender's strategy δ fluctuates around 20. When the attacker is in the 8th stage and the system reaches a steady state, the defender's reward is 1.06, and the defender's strategy δ fluctuates around 20.

5.3.3. Delay Comparison. In order to evaluate the advantages of the MFD-PPO algorithm, this section first compares MFD-A2C with MFD-PPO. Figure 11 shows the training result of MFD-PPO and MFD-A2C. The experimental result shows that although MFD-A2C converges faster than the MFD-PPO algorithm, the reward of MFD-A2C is not monotonically increasing during the training process. Moreover, when MFD-A2C converges, the fluctuation of

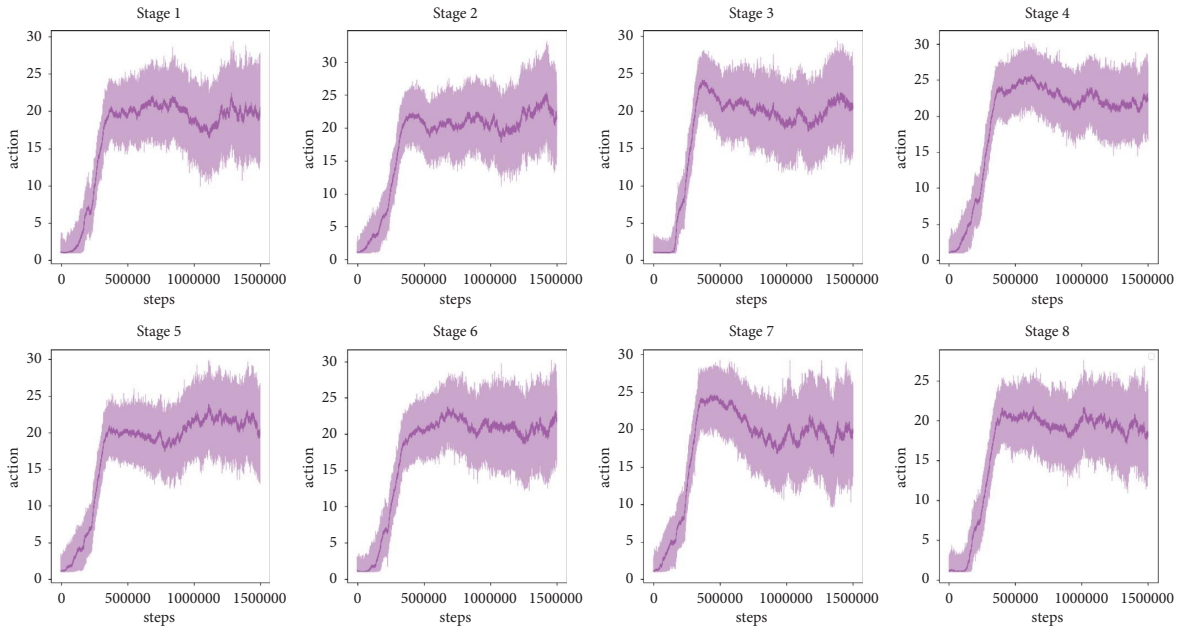

Figure 10: The strategy of defender at each stage.

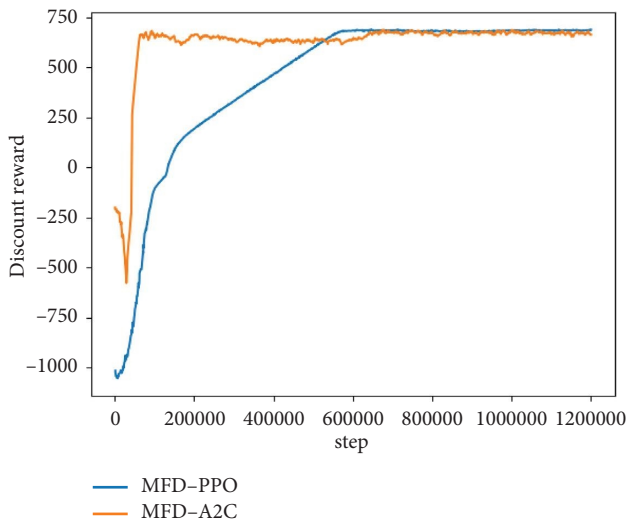

Figure 11: Convergence performance of MFD-A2C and MFD-PPO algorithms.

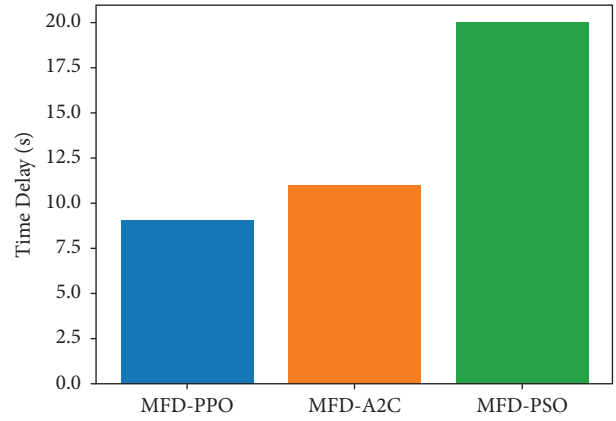

Figure 12: The delay comparison of MFD-PPO, MFD-A2C and MFD-PSO.

reward is far greater than that of MFD-PPO. Therefore, the MFD-PPO algorithm is stable when solving multi-stage deception strategy. In addition, this paper compares the time delay of the MFD-A2C, MFD-PSO, and MFD-PPO algorithms in getting a multi-stage deception strategy. Since the deep reinforcement learning algorithm includes two stages of training and decision-making, it is meaningless to compare the training time delay of deep reinforcement learning with the time delay of PSO. Therefore, we select the decision-making time delay of MFD-PPO and MFD-A2C to compare with the delay of MFD-PSO, the experimental results are shown in Figure 12. It can be seen from the experimental results that the deep reinforcement learning algorithms MFD-PPO and MFD-A2C have obvious advantages over MFD-PSO in decision-making. MFD-PSO needs to recalculate the multi-stage deception strategy when the environment changes, while the MFD-PPO and MFD-A2C algorithms can use the model to quickly give the multi-stage deception defense strategy after the training is completed. Furthermore, from the experimental results, it can also quickly give the multi-stage deception strategy compared with MFD-A2C, further illustrating the advantages of MFD-PPO.

6. Conclusion

In view of the existing problems in the deception decision-making model based on game, this paper presents the deception decision-making model based on multi-stage Flipit game. Firstly, based on the moving attack surface and the deception attack surface, this paper proposes a deception model based on the moving deception attack surface. Then,

on the basis of analyzing the attack and defense behavior of network deception, a single-stage spatial-temporal decision-making deception model is constructed based on Flipit game. On this basis, we present a multi-stage spatial-temporal decision deception game model by introducing discount reward and stage transition probability, and give the utility function of the multi-stage deception model. Finally, we design a deep reinforcement learning algorithm MFD-PPO to obtain the optimal deception defense strategy. The experimental results show that the MFD-PPO algorithm has strong stability when getting the multi-stage deception strategy. Moreover, it has lower delay compared with the heuristic algorithm.

Our work opens up new avenues for future research. In our paper, we consider the existence of a strong dominant strategy for the defender and the attacker's strategy varies within a certain range. It is interesting to study the case when the attacker selects random policies in every stage. For future work, we may study the model in which there are optimal strategies for both attackers and defenders, and propose the deception strategy selection method based on multi-agent deep reinforcement learning. By using the method, the policies of the attacker and defender are random, and neither side knows the policy of the other.

Data Availability

https://github.com/hwz9612/multi-stage-Flipit-game available on Corresponding author request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Weizhen He and Jinglei Tan contributed equally to this work.

Acknowledgments

the National Natural Science Foundation of China (No. 62072467), the National Key R&D Program of China (No. 2021YFB1006200, No. 2021YFB1006201) and the National Natural Science Foundation of China (No. 62002384).

References

[1] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: techniques, solutions, challenges, and research opportunities," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1851–1877, 2019.

[2] Y. Zhang, R. H. Deng, S. Xu, J. Sun, Q. Li, and D. Zheng, "Attribute-based encryption for cloud computing access control: a survey," *ACM Computing Surveys*, vol. 53, no. 4, pp. 1–41, 2020.

[3] S. Shamshirband, M. Fathi, A. T. Chronopoulos, A. Montieri, F. Palumbo, and A. Pescapè, "Computational intelligence intrusion detection techniques in mobile cloud computing environments: review, taxonomy, and open research issues," *Journal of Information Security and Applications*, vol. 55, Article ID 102582, 2020.

[4] L. Zhang and V. L. L. Thing, "Three decades of deception techniques in active cyber defense retrospect and outlook," *Computers & Security*, vol. 106, Article ID 102288, 2021.

[5] J. Pawlick and Q. Zhu, "Deception by design: evidence-based signaling games for network defense," 2015, https://arxiv.org/abs/1503.05458.

[6] J. Pawlick, E. Colbert, and Q. Zhu, "Modeling and analysis of leaky deception using signaling games with evidence," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 7, pp. 1871–1886, 2019.

[7] R. Píbil, V. Lisý, C. Kiekintveld, B. Bosanský, and M. Pechoucek, "Game Theoretic Model of Strategic Honeypot Selection in Computer networks," *International Conference on Decision and Game Theory for Security*, Springer, Berlin, Heidelbergpp. 201–220, 2012.

[8] W. Hofer, T. Edgar, D. Vrabie, and K. Nowak, "Model-driven Deception for Control System environments," in *Proceedings of the 2019 IEEE International Symposium on Technologies for Homeland Security (HST)*, pp. 1–7, IEEE, Woburn, MA, USA, November 2019.

[9] M. O. Sayin and T. Başar, "Deception-as-defense Framework for Cyber-Physical Systems," *Safety, Security and Privacy for Cyber-Physical Systems*, Springer, Cham, Switzerland,pp. 287–317, 2021.

[10] A. Brzeczko, A. S. Uluagac, R. Beyah, and J. Copeland, "Active Deception Model for Securing Cloud infrastructure," in *Proceedings of the 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 535–540, IEEE, Toronto, Canada, December 2014.

[11] H. Li, Y. Guo, S. Huo, H. Hu, and P. Sun, "Defensive deception framework against reconnaissance attacks in the cloud with deep reinforcement learning," *Science China Information Sciences*, vol. 65, no. 7, pp. 1–19, 2022.

[12] M. Qasem and H. M. J. Almohri, "An Efficient Deception Architecture for Cloud-Based Virtual networks," 2020, https://arxiv.org/abs/2004.06933.

[13] H. Li, Y. Guo, P. Sun, Y. Wang, and S. Huo, "An optimal defensive deception framework for the container-based cloud with deep reinforcement learning," *IET Information Security*, vol. 16, no. 3, pp. 178–192, 2022.

[14] O. Tsemogne, Y. Hayel, C. Kamhoua, and G. Deugoué, "Game-Theoretic modeling of cyber deception against epidemic botnets in Internet of Things," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2678–2687, 2022.

[15] M. S. Pour, J. Khoury, and E. Bou-Harb, "HoneyComb: a darknet-centric proactive deception technique for curating IoT malware forensic artifacts," in *Proceedings of the NOMS 2022-2022 ieee/IFIP network operations and management symposium*, pp. 1–9, IEEE, Budapest, Hungary, April 2022.

[16] S. Achleitner, T. F. La Porta, P. McDaniel, S. Sugrim, S. V. Krishnamurthy, and R. Chadha, "Deceiving network reconnaissance using SDN-based virtual topologies," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1098–1112, 2017.

[17] C. Y. J. Chiang, S. Venkatesan, S. Sugrim et al., "On Defensive Cyber Deception: A Case Study Using SDN," in *Proceedings of the MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, pp. 110–115, IEEE, Los Angeles, CA, USA, October 2018.

[18] J. Kim, J. Nam, S. Lee, V. Yegneswaran, P. Porras, and S. Shin, "BottleNet: hiding network bottlenecks using SDN-based topology deception," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3138–3153, 2021.

[19] F. Cohen and D. Koike, "Leading attackers through attack graphs with deceptions," *Computers & Security*, vol. 22, no. 5, pp. 402–411, 2003.

[20] S. Milani, W. Shen, K. S. Chan et al., "Harnessing the Power of Deception in Attack Graph-Based Security games," *International Conference on Decision and Game Theory for Security*, Springer, Cham, Switzerland,pp. 147–167, 2020.

[21] A. Sayari, Y. Djemaiel, S. Rekhis, A. Mabrouk, and B. Jerbi, "Attack Modeling and Cyber Deception Resources Deployment Using Multi-Layer Graph," *International Conference on Advanced Information Networking and Applications*, Springer, Cham, Switzerland,pp. 560–572, 2022.

[22] T. E. Carroll and D. Grosu, "A game theoretic investigation of deception in network security: deception in network security," *Security and Communication Networks*, vol. 4, no. 10, pp. 1162–1172, 2011.

[23] L. Huang and Q. Zhu, "Dynamic Bayesian Games for Adversarial and Defensive Cyber deception," *Autonomous cyber deception*, Springer, Cham, Switzerland,pp. 75–97, 2019.

[24] M. Ahmadi, M. Cubuktepe, N. Jansen, S. Junges, J. P. Katoen, and U. Topcu, "The Partially Observable Games We Play for Cyber deception," 2018, https://arxiv.org/abs/1810.00092.

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

[26] A. Schlenker, O. Thakoor, H. Xu, F. Fang, L. T. Thanh, and E. Vorobeychik, "Deceiving Cyber Adversaries: A Game Theoretic approach," *International Conference on Autonomous Agents and Multiagent Systems*, Springer, Cham, Switzerland, 2018.

[27] Y. Yin, B. An, Y. Vorobeychik, and J. Zhuang, "Optimal deceptive strategies in security games: a preliminary study," 2013, https://arxiv.org/pdf/1905.04833.pdf.

[28] I. Anjum, M. S. Miah, M. Zhu et al., "Optimizing Vulnerability-Driven Honey Traffic Using Game Theory," 2020, https://arxiv.org/abs/2002.09069.

[29] H. Q. Ngo, M. Guo, and H. Nguyen, "Near optimal strategies for honeypots placement in dynamic and large active directory networks," in *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pp. 2517–2519, New York, NY, USA, August 2023.

[30] J. Tan, H. Jin, H. Zhang et al., "A survey: when moving target defense meets game theory," *Computer Science Review*, vol. 48, Article ID 100544, 2023.

[31] J. Tan, H. Jin, H. Hu, R. Hu, H. Zhang, and H. Zhang, "Evolutionary Decision Method for Moving Target Defense Based on Wright-Fisher Process," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, 2022.

[32] S. Dowling, M. Schukat, and E. Barrett, "Using Reinforcement Learning to Conceal Honeypot functionality," *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, Cham, Switzerland,pp. 341–355, 2018.

[33] S. Wang, Q. Pei, J. Wang, G. Tang, Y. Zhang, and X. Liu, "An intelligent deployment policy for deception resources based on reinforcement learning," *IEEE Access*, vol. 8, pp. 35792–35804, 2020.

[34] N. C. Abay, C. G. Akcora, Y. Zhou, K. Murat, and B. Thuraisingham, "Using deep learning to generate relational honeydata," *Autonomous Cyber Deception*, Springer, Cham, Switzerland,pp. 3–19, 2019.

[35] P. K. Manadhata, "Game theoretic approaches to attack surface shifting," *Moving Target Defense II*, Springer, New York, NY, USA, 2013.

[36] Y. Huang and A. K. Ghosh, *Introducing Diversity and Uncertainty to Create Moving Attack Surfaces for Web services*, Springer, New York, NY, USA, 2011.

[37] M. Albanese, E. Battista, and S. Jajodia, "Deceiving Attackers by Creating a Virtual Attack Surface," *Cyber Deception*, Springer, Cham, Switzerland,pp. 167–199, 2016.

[38] D. Ma, Z. Tang, X. Sun, L. Guo, L. Wang, and K. Chen, "Game theory approaches for evaluating the deception-based moving target defense," in *Proceedings of the 9th ACM workshop on moving target defense*, pp. 67–77, New York, NY, USA, December 2022.

[39] C. Lei, H. Q. Zhang, L. M. Wan, L. Liu, and D. Ma, "Incomplete information Markov game theoretic approach to strategy generation for moving target defense," *Computer Communications*, vol. 116, pp. 184–199, 2018.

[40] M. Van Dijk, A. Juels, A. Oprea, and R. L. Rivest, "FlipIt: the game of "stealthy takeover"," *Journal of Cryptology*, vol. 26, no. 4, pp. 655–713, 2013.

[41] Y. Zhang, X. B. Tan, X. L. Cui, and H. S. Xi, "Network security situation awareness approach based on markov game model: network security situation awareness approach based on markov game model," *Journal of Software*, vol. 22, no. 3, pp. 495–508, 2011.

[42] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization algorithms," 2017, https://arxiv.org/abs/1707.06347.

[43] U. Doraszelski and J. F. Escobar, "A theory of regular Markov perfect equilibria in dynamic stochastic games genericity, stability and purification," *Theoretical Economics*, vol. 5, no. 2, pp. 369–402, 2015.

[44] A. Nilim and L. El Ghaoui, "Robust control of markov decision processes with uncertain transition matrices," *Operations Research*, vol. 53, no. 5, pp. 780–798, 2005.

[45] China National Vulnerability Database of Information Security, "China National Vulnerability Database of Information Security," 2020, https://blog.csdn.net/weixin_41686586/article/details/113996298.