









Research Article

Discover the Hidden Attack Path in Multiple Domain Cyberspace Based on Reinforcement Learning

Lei Zhang ^{1,2} Hongmei Li ² Shiming Xia ¹ Yu Pan ¹ Wei Bai ¹ Qin Feng,¹
Wei Li ¹ Qibin Zheng ³ Shize Guo,¹ and Zhisong Pan ¹

¹Army Engineering University of PLA, Nanjing, China

²Academy of Military Science, Beijing, China

³Advanced Institute of Big Data, Beijing, China

Correspondence should be addressed to Zhisong Pan; hotpzs@hotmail.com

Received 22 March 2022; Revised 5 September 2022; Accepted 14 September 2022; Published 10 October 2022

Academic Editor: Liang Zhao

Copyright © 2022 Lei Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

There are certain vulnerabilities at the beginning of multiple domain cyberspace configuration. How to discover these potential vulnerabilities has been a hot topic. This paper proposes to find these cyberspace vulnerabilities by discovering the shortest attack path in multiple domain cyberspace. In order to discover more and shorter attack paths, we train an agent as an attacker to discover multiple domain cyberspace attack paths. We formulate the discovering attack paths as a reinforcement learning (RL) problem. With this technique, we added a multiple domain action select module to RL that can pick an executable action in a state. By using the proposed method, we can discover more hidden attack paths and shorter attack paths to analyze the potential vulnerabilities to cyberspace. Finally, we created a simulated cyberspace experimental environment to test our proposed method. The experimental results show that the proposed method can discover more hidden multiple domain attack paths and shorter attack paths than the existing methods.

1. Introduction

Amazing improvements in computer hardware and software have permitted the rapid advancement of artificial intelligence (AI), yet it is not frequently employed to defend cyberspace security [1]. However, a significant issue with the existing cyberspace security defense is the intellectualization of that system. Cyberspace should truly be viewed in terms of a place made up of physical, digital, network, and social domains when it comes to managing its security. Its security should also be managed as a whole [2]. This procedure primarily entails the intelligent deployment of security devices and the intelligent discovering of hidden attack paths [3], intelligent monitoring of network traffic, and intelligent awareness of security situation and other parts [4], which comprehensively constitute the cyberspace security protection system. We believe that it is AI itself that will provide the means to constitute the cyberspace security protection system, creating a symbiotic relationship between cyberspace security and AI with each fueling advances in the other.

Here, we proposed a learning-based method for discovering hidden attack paths in multiple domain cyberspace. In order to effectively fix cyberspace flaws and increase the security of the multiple domain cyberspace, our goal is to identify the shortest hidden attack path. We define the weakest vulnerability in multiple domain cyberspace as the shortest hidden attack path. Even if this issue has been researched, it still depends on specialists to determine and assess the security dangers present in the current cyberspace, which is not very intelligent and requires a lot of human resources. Additionally, only network domain may be taken into account in the current research since the problem complexity results from the interplay of numerous domains in cyberspace. Physical, social, network, and digital domains should all be present in the multiple domain cyberspace, in general. Space and physical objects are included in the physical realm. City, campus, building, and room are all examples of space domain. Physical domains include switch, router, computer, and other terminal and network equipment. Social relationships are referred to as the social

domain. For instance, if an attacker is friends with the network administrator, it will be simpler for them to get greater network rights than other attackers. The typical network area is referred to as the network domain. Information entities used to represent digital information, such as user names, passwords, secret keys, communications, are known as digital domains. This results in the great complexity of discovering hidden attack paths due to the interplay of multiple domains in cyberspace. As a result, discovering the shortest attack path cannot be done exhaustively. The state space is still so huge that the problem cannot be solved using conventional techniques even after it has been divided into smaller problems.

According to our knowledge, the following issues must be resolved in order to achieve the goal.

The first is the issue of multiple domains that can interact with one another and how to adjust the current analysis of cyberspace security risks to emphasize numerous cyberspace domains and enforce the pertinence and relevance of business level.

The second is the issue of alternative actions in various states being unique. The number of actions an agent may choose from in classical reinforcement learning (RL) is constant throughout all states. However, the agent has a variety of possible actions in a variety of states. Different approaches must be used to this problem's solution in order to make the alternative actions in various states.

To address this challenge, we treat discovering the hidden attack paths across multiple domains as an RL problem and train an agent to discover the hidden shortest attack path over multiple domain cyberspace. The RL agent successively discovers the attack paths throughout each training cycle. A quick yet approximate reward directs training so that each RL agent may learn the attack paths. In addition, we proposed an enhanced Deep Deterministic Policy Gradient (DDPG) method to enable the agent to choose various actions in various states.

We think our method opens up new opportunities for network administrators since it can grow over time and learn from experience. The experimental results demonstrate that the proposed method is capable of outperforming baseline methods in a simulated cyberspace. Additionally, our proposed method can discover shorter attack paths at the same time as other baseline methods. Our proposed method is extensively applicable to many genuine cyberspaces, even if we assess largely on a simulated cyberspace.

The following are this paper's key contributions.

We suggest using RL to the management and upkeep of cyberspace in order to discover hidden attack paths across multiple domain cyberspace. Our experimental results demonstrated that cyberspace is less safe when the hidden attack path is shorter. We may identify cyberspace weaknesses using the way we have suggested, which will help administrators by serving as a guide and enhancing the ability of cyberspace security to operate and be maintained.

We proposed an enhanced DDPG algorithm to handle the issue of diverse alternative actions in various states. We can determine if an action can be executed in a state and acquire an execution action in a state by providing a multiple

domain action selection module. Through this method, we may realize the choice of actions in various states.

This paper is organized as follows: introduction is in Section 1. We talk about the related works in Section 2. Our proposed method, which includes the enhanced DDPG algorithm and model for discovering hidden attack paths, is presented in Section 3. Our experiment and an analysis of the experimental results are presented in Section 4. The conclusion is in Section 5.

2. Related Works

2.1. Intelligent Security Protection. The primary focus of intelligent security protection is user's behavior and network monitoring guidelines. The traffic will often alter as a cyberattack manifests, and we may use the attack mode type to our advantage to find abnormalities in the cyberspace [5]. We may gather the original message of the data in the cyberspace and extract it, take the destination address and other information, construct the typical traffic model, and then apply discrete wavelet transform technology to analyze and identify the data traffic in order to find abnormalities in cyberspace [6].

Today's cyberspace user action-based intelligent security protection method mostly depends on online data mining, user abnormal activity detection, and technique based on neural networks to discriminate. In this study, such covert attack paths are referred to as zero-day attack paths, and a prototype system called Patrol is proposed to detect them in real time [7]. But because this method is dependent on rules, it is less effective because it requires professionals to establish the rules. Many current methods have the drawback of not scaling well to medium-sized networks with hundreds or thousands of hosts; therefore, this study introduced graph reduction techniques to make the effort of locating and removing the best attacker pathways simpler [8]. Additionally, it offered a way to prioritize both individual vulnerability and attack vectors in another study [9]. From a different angle, a software protection meta-model is suggested that may be used to create a formal knowledge base that has just that information. Real-time analysis of cyberspace security warnings can be provided using this method [10].

Users will remember a lot of action information throughout operation, and making efficient use of this knowledge is the foundation and key to realizing the determination of abnormal action. To assist the choice of user access action, multiple layer log collecting is built. Using this method, we may identify attacker's activity. The methods above can determine whether a user's behavior is abnormal or not by using a multilevel user access log, integrating web front, user click action, and URL access logic, extracting the user's access action characteristics, and then computing average user behavior baseline characteristics, using an effective monitoring abnormal access action scoring algorithm, and tracing the action of the abnormal IP [11].

The neural network method has been effectively used in the domains of pattern recognition and probability density estimation as a crucial technique for dealing with nonlinear

systems. The aberrant behavior analysis approach based on neural network can better describe the nonlinear relationship between one variable and another when compared to the statistical analysis theory. The capacity of the behavior analysis system to examine a large number of network packets is necessary for the changing of aberrant network behavior. Furthermore, several attackers may coordinate a number of common attacks, necessitating the need for an intelligent security defense system with the capacity to handle a sizable volume of nonlinear data. The neural network-based method offers a high degree of flexibility for the study of intelligent security protection because of its quick reaction time, particularly for the processing of noisy and incomplete data [12].

Intelligent security protection has recently become popular due to the development of machine learning. SVM [13, 14], K-nearest neighbors [15], Naive Bayes [16], random forests [17], neural networks [18], deep learning, and other novel methods are only a few of the recent attempts. Due to their superior performance, deep learning-based techniques have entered the mainstream in the industry. Gao proposed a deep belief network model that employs a supervisor-based back-propagation network with a multilayer unsupervised learning network [19]. In order to learn network traffic characteristics unsupervised, Shone employed asymmetric depth self-encoders, which not only produced good results on big data sets but also cut down on training time [20]. Recurrent Neural Network (RNN) was the model that Yin proposed, compared to the nondepth model's efficacy, and successfully executed [21]. Long Short-Term Memory (LSTM) and a gradient descent approach were used in Kim's model [22]. The outcomes of the trial demonstrated that the LSTM can perform better. Sheraz did a thorough investigation into the deep learning model and showed that it can be applied not just in this area but also to provide superior results [23].

2.2. Reinforcement Learning and Cyberspace Simulation. Reinforcement learning is commonly considered as a general machine learning model. It mainly studies how agent can learn certain policy by interacting with the environment, to maximize long-term reward. RL is based on the Markov Decision Process (MDP) [24]. A MDP is a tuple (S, A, T, R, γ) , in which S is the set of states and A is the set of actions. $T(s_i|s_j, (a))$: $T \times A \rightarrow R$ is the reward after executing action a at stage s_j , and γ is the discounting factor. We used π to denote a stochastic policy, $\pi(s, a)$: $T \times A \rightarrow [0,1]$ is the probability of executing action a at state s and $\sum_{a \in A} \pi(s, a) = 1$ for any s . The goal of RL is to find a policy π that maximizes the expected long-term reward. Besides, the state action value function is

$$Q^\pi(s, a) = E \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_t \sim \pi(s_t) \right], \quad (1)$$

where $\gamma \in (0, 1]$ measure the importance of future reward to current decisions.

They indicate the potential for multiple actions to be chosen in the same state for various policies π , and they also

correlate to various rewards. In the same state, a superior policy can choose better action to reap more rewards.

Traditional RL uses methods like Dynamic Programming, the Monte Carlo Method, and Temporal-Difference Learning to interactively compute the action value function, which finally converges to provide the best policy. Following the proposal of deep learning, the deep reinforcement learning approach created by merging RL is currently the most used method.

We present the popular RL algorithms A2C and DDPG in the sections that follow.

Advantage Actor Critic (A2C): A2C is similar to Asynchronous Advantage Actor Critic (A3C) [25] but has no asynchronous part. The advantage function, which may be used to assess the quality of the selected action values and the averages of all actions, is employed by A2C in place of the critic network's raw results. Algorithms are being learned by A2C and A3C exploiting the advantage. The following equation represents the benefit: $A(s, a) = Q(s, a) - V(s)$. The advantage is denoted as $A(s, a)$. We denote the state of the agent as s and the action as a . $V(s)$ represents the pure value of the state s ; therefore, $A(s, a)$ represents the pure value of action a . Previous research has shown that the advantage can solidify learning. A2C performs better than A3C while lacking the asynchronous component. Thus, in order to maintain and enhance the learning process, we adopt A2C as a baseline in our research.

Deep Deterministic Policy Gradient (DDPG): DDPG [26] is a learning method that integrates deep learning neural network into Deterministic Policy Gradient (DPG) [27]. Compared with DPG, the improvement is using neural network as policy network and Q-network and then used deep learning to train the above neural network. DDPG consists of four networks: actor current network, actor target network, critic current network, and critic target network. In addition to the four networks, DDPG also uses experience playback, which is used to calculate the target Q-value. In Deep Q-Network (DQN), we copy the parameters of the current Q-network directly to the target Q-network, that is, $\theta^Q = \theta^Q$, but DDPG uses the following update:

$$\begin{cases} \theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}, \\ \theta^{\mu'} \leftarrow \tau \theta^{\mu} + (1 - \tau) \theta^{\mu'}, \end{cases} \quad (2)$$

where τ is the update coefficient, which is usually set as a small value, such as 0.1 or 0.01. And this is the loss function:

$$L(w) = \frac{1}{m} \sum_{j=1}^m (y_j - Q(\phi(S_j), A_j, w)). \quad (3)$$

RL has several uses in a variety of contexts, particularly in engineering. A multiagent safe RL method is used to take into account the system's security management as well as the mutual effects of numerous agents [28]. A bilevel RL model for security management of the system is created [29], which can also adaptively learn from prior performance and enhance it over time. DDPG and A2C share many similarities, but they utilize the critic network in different ways. Using solely critic is merely an alternative to increase stability in

A2C since we utilize critique as a baseline from an experience trajectory. This is so that we can avoid back-propagation in A2C when the policy is stochastic. Because the deterministic nature of its policy allows us to calculate the gradient from the Q -value, critique is utilized differently in DDPG. The critic network, which in turn employs the actions produced by the actor network, provides the Q -value. Because DDPG is a deterministic policy RL algorithm, we choose to employ it in our work and develop it. Its benefit is that efficiency has improved. In order to determine the precise value of action at each step, we also need to sample the probability distribution of the best course of action, but in A2C, action is often a high-dimensional vector with between 25 and 50 dimensions. Undoubtedly, frequent sampling in the high-dimensional action space uses up computer resources. Furthermore, because agents in this cyberspace environment might take varied actions depending on their state, DDPG is not only more effective but also better suited to it. Finally, to show the superiority of the proposed method, A2C and DDPG were compared to it in our trials.

In parallel, a game is being mimicked online. Two agents, an attacker and a defender, compete in an adversarial sequential decision-making game. Cyberspace simulation is represented as a graph network made up of nodes where attackers and defenders can move [30]. The node where an attacker or defender is located determines their condition. The exploit is successful when the attack value of each action taken by the attacker is greater than the defensive value of the defender. This method's drawback is that the two agents used RL against one another and tested how well they performed against opponents who were learning. This study demonstrated that we can model an environment to replicate cyberspace.

However, there are not many environments like the one from the prior job. In this situation, both the attacker and the defender will adjust and interact in real time. However, after the attackers succeeded in 62 out of their assault objectives, their attacks were discovered [31]. As a result, real-time competition between attackers and defenders is uncommon. In this research, we train the attack agents in the absence of the defensive agents in cyberspace settings.

3. Proposed Method

The problem definition: an overview of how the problem is formulated as an RL problem and a full explanation of the reward function, action, state, policy, and policy updates are all covered in this part. Finally, we presented our enhanced DDPG method to address this issue.

3.1. Overview. The proposed method, that is, the elements of the reinforcement learning agents' training, is described in this section. We start by defining the agent's state s . Second, we provide the course of action a that the agent chose to take in the cyberspace (how to select from the action lists). Third, we determine rewards r based on the outcome of the action a . Finally, we proposed our method for enhancing DDPG

algorithm in this multiple domain cyberspace. The agent continues learning by accumulating the set of s , a , and r seen from the learning environment. An overview of our model is shown in Figure 1.

3.2. Problem Definition. The goal of this study is to identify the shortest hidden attack paths in multiple domain cyberspace in order for the administrators to identify the cyberspace's vulnerabilities and take action to strengthen cyberspace security. This is how the issue is described: Figure 1 illustrates how we saved the security information in S2 given a cyberspace environment. The attacker cannot access the S2 and obtain the security information under the current cyberspace setup. Our goal is to use RL to train an attacker agent to get access to the S2 and obtain security information. If an attacker is able to access S2 and obtain the security information, the assault was successful.

We treat the problem as a MDP, that is, $M = (S, A, P, R, \gamma)$, where $s \in S$ is the current state of the cyberspace, $a \in A$ is an attack action that is currently available, P is the probability of transitions between states, R is the reward value after taking an action to reach the next state, and γ is the discount factor. For the transfer probability, it can be expressed as $p(\hat{s}|s, a) = p(S_{t+1} = \hat{s} | S_t = s, A_t = a)$. For the reward function, it can be formally expressed as $R(s, a) = E[R_{t+1} | s, a]$.

An RL agent acting as an attacker and a configured cyberspace are present in our settings at the starting state s_0 . The final state s_t reflects whether an attacker launched a limited-step assault successfully or not. The RL agent will execute an action to finish an attack step at each stage. As a result, T is the quantity of attack steps in total. The agent starts off in state s_t , takes action a_t , moves to state s_{t+1} , and receives a reward from the cyberspace environment r_t at each step t .

We define s_t as a concatenation of characteristics that describe the state at time t , such as the agent's location, the computer they are using, and the services they have access to. In Section 3.4, we will go into more depth about it.

The following state, s_{t+1} , has an updated representation with details about the attacker's location at the time of the attack or how they acquired the new service information.

The agent will do any legitimate actions in the state s_t that are in the action space. The agent's state space, or a_t , is s_t that the agent selected. In Section 3.5, we will go into more detail about it.

The shortest attack path will be discovered in this study, subject to restrictions on cyberspace security setup or security equipment. A fixed value divided by the total number of attack steps represents our final reward. We shall discuss it in full in Section 3.6.

3.3. Definition of Multiple Domain Cyberspace. More and more academics are realizing that diverse domain behaviors have an impact on cyberspace as their understanding of cyberspace. Cyberspace should be understood as being integrated into multiple domains, including the physical, information, network, and social ones. It should also use an

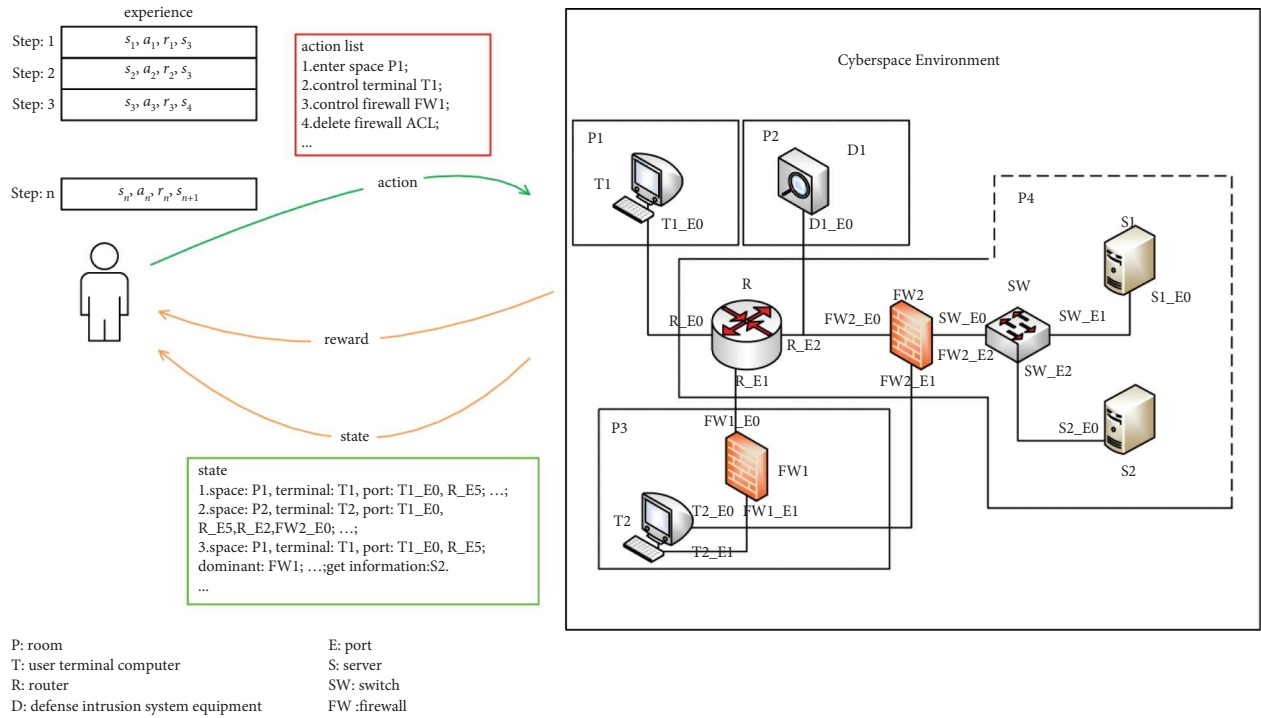


FIGURE 1: The overview of proposed model.

interconnected information technology infrastructure network as a platform, transmit data via radio and cable channels, and regulate the actions of entities, with a focus on its multiple domains. The social domain attribute of multiple domain cyberspace is not explored in this study since it primarily concerns the social interaction between the network administrator and the attacker. We define cyberspace in this paper as having multiple domains, including the physical, information, and network domains. The equipment’s spatial information is primarily described by the physical domain. The interface, route, and action involved in network transmission are primarily described by the network domain. The majority of the digital information in cyberspace is referred to as the information domain.

Additionally, there are some security regulations in this cyberspace, such as laws for the protection of the physical domain, the network domain, and the information domain. Physical domain security protection rules primarily outline the ways to prevent unauthorized employees from entering a certain area and unauthorized access to the physical domain. The rules for network domain security protection primarily outline the ways to stop unauthorized access in the network domain. Network isolation may generally be achieved through the use of Access Control Lists (ACLs), static routing, VLAN partitioning, and other techniques. ACLs are stated as permitting data to flow through a port at a source address, a port at a destination address, and a service at a destination address. The techniques to stop unauthorized access in the information domain are mostly covered by the regulations for protecting the security of that domain. The fundamental technique is to encrypt the data before it is stored or sent. A secret key is required to decrypt the file

regarding whether it was encrypted using public key or symmetric cryptography.

We address the challenge of discovering hidden attack paths using a deep RL algorithm, using an RL agent to discover attack paths across multiple domain cyberspace. When the RL agent successfully attacks the service, we will reward them positively. The ultimate reward is equal to the award divided by the number of attack sequence steps. We will give the RL agent a negative reward if they fail to attack the service or use more attack steps than allowed. RL issues may be expressed as MDPs, which have three essential components: states, actions, and rewards. We define each of the items listed below.

3.4. Definition of Agent’s State. States: the location of the RL agent, the device he is using, and the device permissions he possesses are all included in the collection of potential states of the multiple domain cyberspace. The key component of cyberspace states is the attacker’s permissions, which he might acquire through a sequence of activities. In this paper, we go through nine different sorts of attacker permissions, including Space-Enter, Object-Use, Object-Dominate, Port-Use, Port-Dominate, Service-Reach, Service-Dominate, File-Dominate, and Information-Know. A physical domain’s permissions are Space-Enter, Object-Use, and Object-Dominate. Space-Enter means an attacker enters a physical space, Object-Use means an attacker is permitted to use a device, terminal, or piece of equipment, and Object-Dominate means an attacker controls an object. Attackers can control a piece of property by using an object. The distinction between Object-Use and Object-Dominate is that

the former can only utilize the equipment in its present condition, whilst the latter can modify the equipment. The network domain's permissions are Port-Use, Port-Dominate, and Service-Reach. When a port is used, the attacker can access the network. Attackers who have control over a port can alter its status or configuration. Service-Reach denotes the information flow necessary to fulfill a service request but prevents a user from accessing the service. Via using the secure authentication service, the attacker can access the service by File-Dominating. The digital domain's permissions are File-Dominate and Information-Know. Attacker who controls a file can read, delete, and alter it, among other things. Information-Knowledge refers to how an attacker learned about security, for example, administrator user name, administrator password, and administrator key.

In this study, the RL states are used to control the attacker's access to cyberspace. We established a state list that covered the attacker's device permissions. For instance, if an attacker is in a space, the following values are set for the attacker's state: Room A Space-Enter=0, Room B Space-Enter=0, Terminal A Object-Use=0, Terminal A Object-Dominate=0, Port B Port-Use=0, Port B Port-Dominate=0. The attacker then takes the following action: enters Room A. This causes Room A Space-Enter to be set to 1 and all other states to be set to 0. When an attacker takes action, the states change as a result of the attacker's action.

3.5. Definition of Agent's Action. Actions refer to the collection of steps an agent can take to alter the states of the cyberspace environment (e.g., enter a room, operate or control a computer, and access a service by a port).

The attacker's options are constrained in this cyberspace environment because the action space is so vast. This is distinct from typical RL algorithm, for instance, if the attacker is in outer space and has only two options: "enter room" or "remain motionless." He is not given the option to "operate a computer," "manage a computer," or perform any other operations. As a result, we will impose certain restrictions on the states when an attacker chooses an action. We will go into more depth about this in Section 3.7.

3.6. Definition of Agent's Reward. Reward is the incentive for an agent to behave in a state.

The reward setting is described in this section. The agent's objective in our tests for Section 4 is to retrieve the security information from S2 and the key to decode the data. First, the attacker must figure out how to use and dominate S2 objects in order to gain the password needed to decrypt the security information. If the attacker is able to obtain the Information-Know permission to access the security information, the attack is successful. Due to the multiple domain cyberspace security rules, the attacker cannot access server S2 directly. Instead, the attacker must access FW2 to change the ACLs, allowing him to access servers S1 and S2. If the attacker is successful in obtaining the security information, he can then access server S2 and change the ACLs

once more to return to the previous state. As a result, the setting for reward r is as follows:

- (i) $r = 5000000/t$ if the attacker obtains the Information-Know of security information in S2, and the t is the attacker's successful sequences steps.
- (ii) $r = 10$ if the attacker obtains the File-Dominate of the security information in S2.
- (iii) $r = 5$ if the attacker obtains the Object-Dominate and Service-Dominate of S2.
- (iv) $r = 5$ if the attacker obtains the Object-Dominate and Service-Dominate of S1.
- (v) $r = 1$ if the attacker obtains the Port-Use of S1.
- (vi) $r = 1$ if the attacker obtains the Port-Use of S2.
- (vii) $r = -0.1$ if the attacker has an action but obtains no permissions.

Reward $r = -0.1$ represents the agent's penalty. Maximizing the reward is the agent's goal. As a result, the agent will attempt to attain the objective as fast as feasible if $r = -0.1$ is specified for each action. This relates to an attack scenario where the attacker completes the objective as quickly as feasible to reduce the number of attack sequence steps. Additionally, the episode is over once the attack sequences reach 10,000 steps.

3.7. Improved DDPG Algorithm. Our model's primary purpose is to identify an RL agent's hidden attack path in the context of a certain cyberspace setting.

The model uses the DDPG method, and the attacker is represented as an agent. The agent chooses an action in current state, which has the potential to alter both the environment and the agent's state, as the initial step in determining the shortest hidden attack path. Agents will simultaneously get positive or negative rewards. Additionally, the agent's state has changed, allowing it to take additional activities to earn greater rewards. As a result, the agent learns by trial and error the hidden attack path in this cyberspace environment. In Figure 2, the model is displayed.

Discovering the appropriate policy mapping function $R(s) \rightarrow A$ is another phase in the process of finding the shortest attack steps in order to choose the action a that would maximize the agent's long-term reward given the present state s of the cyberspace. The policy in this process may be separated into two categories: deterministic policy and stochastic policy. Deterministic policy is for the state and the belief that the output will be commensurate (action). The efficiency of deterministic policy algorithms is often great, but they cannot be explored or improved. Instead of using a deterministic policy, a stochastic policy joins a matching random value, giving it a certain capacity for exploration. In our solution, a deterministic policy is selected to assure superior performance because the action value range in actual applications is typically not big.

It is an ordinary RL model. The agent will behave and get a reward as a result of the study of environmental awareness. The agent's objective is to maximize rewards, which leads to additional agent training. In this research, we will enhance

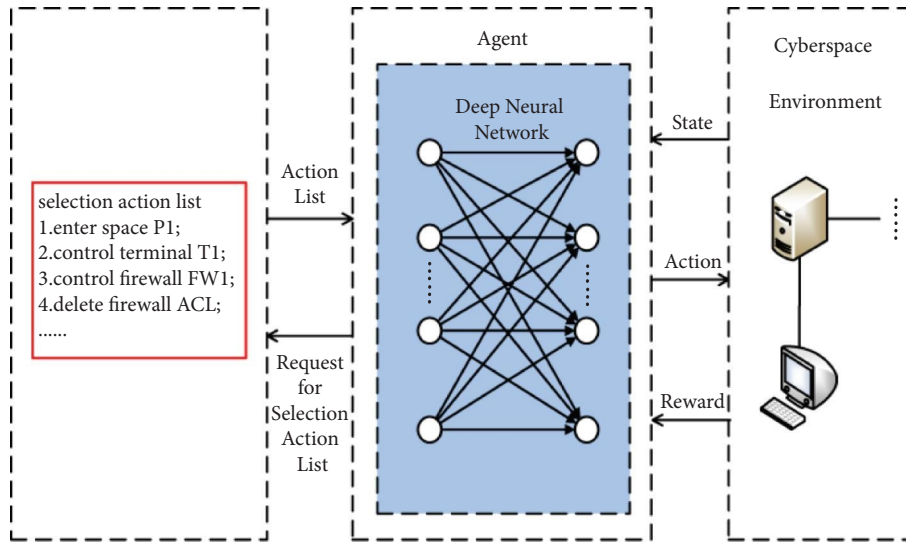


FIGURE 2: The overview of agent discovers the hidden attack path model.

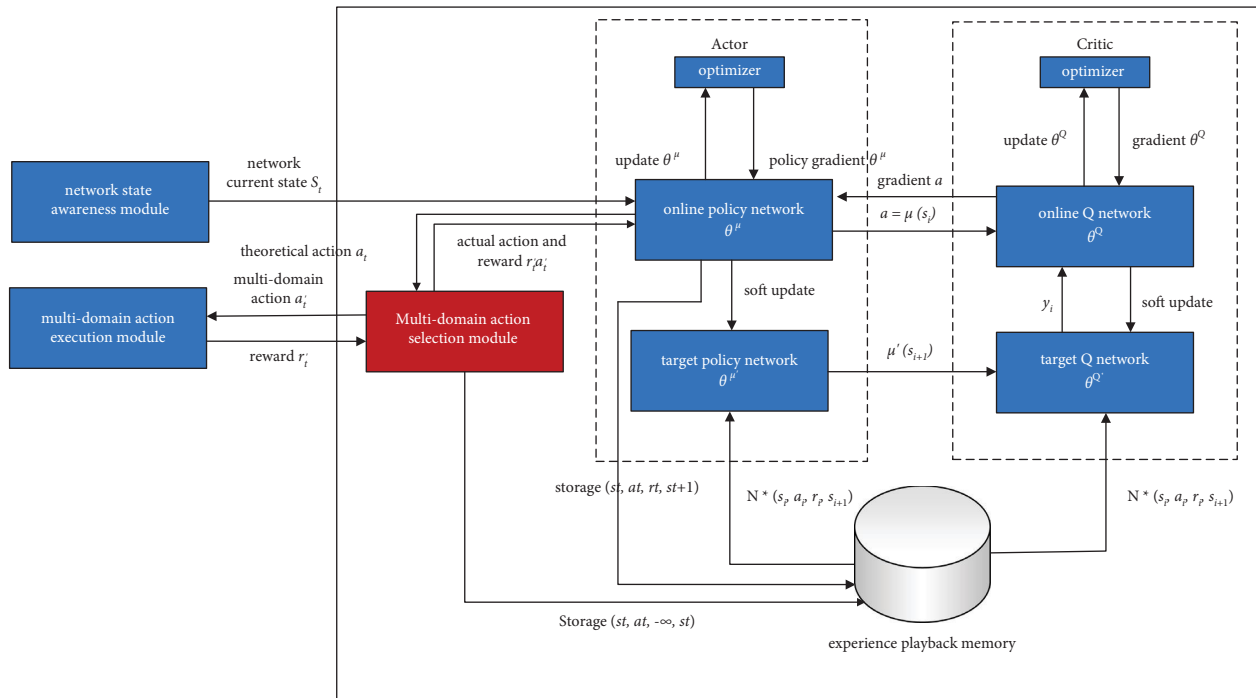


FIGURE 3: The overview of improved DDPG algorithm.

the DDPG algorithm, whose primary design is depicted in Figure 3.

Additionally, there are four networks and one experience replay memory in the conventional DDPG method. With regard to these, the experience replay memory is primarily in charge of storing the state transfer process of $\langle s, a, r, s' \rangle$. Using small batch sampling, the corresponding transferred samples are extracted and used to train the corresponding neural network in order to prevent a high degree of correlation between samples. The four networks are the online policy network, the target policy network, the online Q-network, and the target Q-network. There are two policy

networks (Actor) and two Q-networks (Critic) among the four networks. The deep neural network used by the policy network, which uses the current state as input and produces the appropriate action as the output, is primarily used to replicate the attacker's policy. The basic purpose of the Q-network is to calculate the expectation of the final reward value that would be acquired if the policy were to be continually carried out after the present action had been carried out in a certain condition. Current state and action are the inputs, and the Q-value is the output. The learning process is unstable when a single neural network is employed to replicate a policy or Q-value. Since the target network is used

to calculate the training goal in the DDPG algorithm, policy network and Q -value network create copies of two networks, referred to as the online network and the target network, respectively. After a short while, the model of the online networks' parameter updates to the target networks, making the training process stable and simple to converge.

The standard DDPG algorithm has been enhanced, and it differs from the standard DDPG algorithm in two ways.

We included the multiple domain action selection module in the enhanced DDPG algorithm.

The major difference from the normal DDPG algorithm is the addition of the multiple domain action selection module. The activities that an agent can select in each state are the same in a standard DDPG. However, in this setting, the attacker has a variety of possible actions in each condition when choosing the attack paths in cyberspace. For instance, if the attacker is in outer space, his only available actions are "enter room" and "remain motionless," but in the standard DDPG algorithm, he has access to all available actions. It follows that an attacker in outer space cannot choose the action "control terminal." It joined the multiple domain action selection module to enable the DDPG algorithm to pick various activities in various states. The output of the online policy network is the input for this module. We named this the theory action a_t . Next, we performed the real action a'_t , which was then executed into multiple domain action execution modules and received the relevant reward r'_t . In the end, online policy network receives the corresponding actual execution of the action a'_t and the corresponding reward r'_t . This method makes it possible to realize the rational action selection in many states.

Second, experience playback memory input differs from other input.

The input of the experience playback memory is increased to store the sequence (s_t, a_t, r_t, s_{t+1}) , which is the execution of the action a_t in the state s_t , acquisition of the reward r_t , and conversion to the next state s_{t+1} , in order to ensure that the multiple domain action selection complies with the constraints of the actions on the state. Furthermore, it is prevented that the policy network selects an action that is impractical in the state since the corresponding relationship between state and action must be taken into account while choosing the action. Therefore, it is not sufficient for the multiple domain action selection module to simply apply a linear transformation to change an inoperable action a_t into a viable action a'_t when the policy network picks it in the state s_t . In order to prevent relevant actions from being chosen during the policy network training process, relevant action sequences $(s_t, a_t, -\infty, s_{t+1})$ must be taken to show that actions a_t are executed in the state s_t , the subsequent state obtained is still s_t , and the reward at this time is a huge negative value.

Two policy networks share the same network design, with the state of the network as the input and the action to be chosen as the output. A RNN hidden node is introduced structurally between the hidden layer and the original DDPG input layer. The restructured policy network is composed of five levels. The input layer is the first layer. The RNN hidden layer, which has 32 GRU nodes, is the second layer. 48 fully

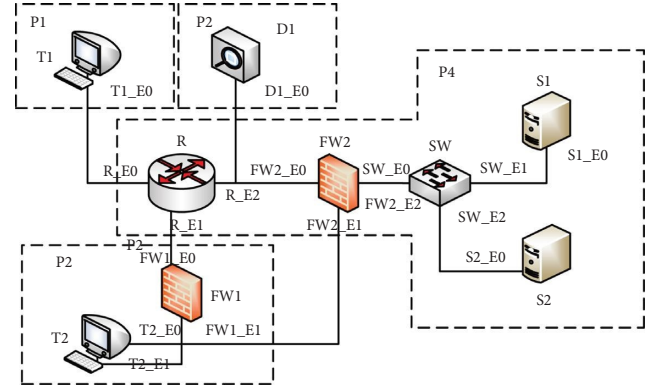


FIGURE 4: Experiment environment topology.

linked nodes are included in the third and fourth levels, which are fully connected layers. The output layer, which is the fifth layer, uses the sigmoid function as the activation function. In the end, it outputs a multidimensional vector that represents the action that has to be taken across multiple domain cyberspace.

The two Q -networks then have another design, whose output is a scalar and whose input comprises both the network's state and a multidimensional vector that represents the matching multiple domain actions. There are four tiers in the network. The input layer is the first layer. There are 48 completely linked nodes in the second layer and 36 in the third layer, respectively. The ReLu function is utilized by the activation function. The output layer, which outputs a scalar and makes use of the linear function as the activation function, represents the Q -value of the appropriate state and action. It is the fourth layer.

4. Experiments

4.1. Experiment Environment. To test the efficacy of our method, we used a model of the cyberspace environment as our experiment data. There are a total of five places in this experiment setting. The entire physical universe, which represents an area, makes up the outermost space. P1 is the location of the terminal, P2 is where the VPN equipment is, P3 is where the communication team is, and P4 is where the communication hub is. There are 5 different types of equipment: computer (T1 and T2, respectively, stored in P1 and P3), firewall (FW1 and FW2, respectively, stored in P3 and P4), sensor (D1, stored in P2), router (R, stored in P4), switch (SW, stored in P4), and server (S1 and S2, stored in P4). Figure 4 depicts the equipment connection relationship. S2 houses the security information. An attack is successful if the attacker can access S2 to retrieve the security file and its "Information-Known" permission. Additionally, we want to make sure that the attack sequence has the fewest steps possible as well as success.

Due to the necessity for remote control of the firewall FW1 equipment in this environment, the FW1_password is still stored in FW1. At the same time, because T2 maintains FW2 and S1, T2 also stores the FW2_password and S1_web_password. The network domain security protection

TABLE 1: Network services in the experiment environment.

Web service	Web service's role	Service support equipment	Service dependent port	Service password
T2_manager	Remote management equipment	T2	T2_E0	None
FW1_manager	Remote management equipment	FW1	FW1_E2	FW1_password
FW2_manager	Remote management equipment	FW2	FW2_E2	FW2_password
S1_web	Web services in server S1	S1	S1_E0	S1_web_password
S2_web	Web services in server S2	S2	S2_E0	S2_web_password

rules only let T1 access server S1 to block other traffic in this cyberspace, while the physical domain security protection rules prohibit users from entering rooms P2 and P4. The security information is kept in S2 and must be decrypted by the user using a password. Since the password is also stored in S2, the user must have “Information-Know” access to the password in order to use it. The attacker can only access S1 in this setup of cyberspace, and they are unable to view S2’s security information. Table 1 lists the network services used in the experiment’s cyberspace environment.

But we do know that an attacker might use a multiple domain joint attack to access the security information kept on server S2 and decrypt it. A possible attack path is as follows.

Attacker first enters space P2 and obtains the firewall FW1_password management service password.

Second, the attacker adds an access control list that allows T1 or D1 to access the management service of T2, which is T2, by using device T1 or D1 to access the management service of FW1_manager.

Third, get the password FW2_password of firewall FW2 stored on T2 and the password S1_web_password of service S1_web through T2_manager.

Fourth, use T2_S1 port, access firewall FW2_manager, add access control list: allow T1 or D1 access service S1_web and S2_web.

Fifth, use T1 or D1 to access the service S1_web and get the password S2_web_password of S2_web.

Finally, the attacker can access the service S2_web using T1 or D1 in order to obtain the security file using S2_web_password. The attacker now completes all of his attack procedures.

This is only one of numerous combined multiple domain attack paths in this cyberspace experiment environment, demonstrating the setup vulnerability of the cyberspace environment. In this work, we propose to detect the weakest point in cyberspace and the shortest hidden attack path, which may serve as a guide for the administrator to correct the weakest in the cyberspace.

4.2. Experiment Method. An agent (attacker) is added throughout the experiment and is first placed in the cyberspace environment depicted in Figure 4.

The attack path sequence steps exceeding 10000 steps is the requirement for each episode’s termination. We also train 500 episodes. If an attacker makes a successful attack, the attack sequence steps n will be recorded, and he will return to the outermost space and continue searching for the attack sequences until the episode has more than 10,000 attack path sequence steps.

We define the related state, action, and reward. In accordance with the RL model and enhanced DDPG algorithm, the following settings are pertinent.

In constructing a state vector, we set a length of 106 vectors on the set of states to represent a state of various positions on the value of the attacker, depending on the spaces, ports, services, or information. For instance, the value for a certain space is set to 1 if the attacker is in it, and to 0 otherwise. Put the value for the port to 1 if the attacker may use it; else, set it to 0. Put the value corresponding to the service to 1 if the attacker is connected to a service; otherwise, set it to 0. We set the value corresponding to that information to 1 if the attacker successfully obtains security information, else to 0.

Attackers have different actions in different states. The introduction is found in Section 3.5. For instance, the attacker can add the necessary access control list for the firewall FW1 in the present state if he has authority over the management service FW1_manage. He is unable to add the FW1 access control list otherwise.

Different rewards are established for the attacker depending on how much of the attack path has been successfully completed. The reward for one of them is determined in Section 3.6 of the program.

4.3. Baselines. We provide three comparative RL methods to test the advantageous of our proposed method. The three RL methods are the following.

DQN method: DQN is an RL standard algorithm that predicts Q -value using a neural network and iteratively updates the neural network to discover the shortest attack path. In DQN, there are two neural networks. The target network, which is comparatively fixed, is used to determine the value of Q -target, and the evaluate-network is used to determine the value of Q -evaluate.

A2C method: as we explained in Section 2, A2C is an RL standard algorithm. Although there is no asynchronous component, this approach is comparable to A3C.

DDPG method: in 2, we presented DDPG, another RL standard algorithm.

The method proposed in this paper is the “improved DDPG method” (IDDPG).

4.4. Results and Discussions. Learn 500 episodes under the aforementioned circumstances, and we noted the following findings.

Result A: in order to compare the average successfully executed attack sequences steps, we added certain security criteria. To do this, we totaled up all of the executed attack

TABLE 2: Attack sequences steps in different security rules.

Methods	3	4	5	6
DQN	3533	4323	7231	8313
A2C	3222	4213	7332	8812
DDPG	2434	4765	7240	8551
IDDPG	995	3322	6321	7233

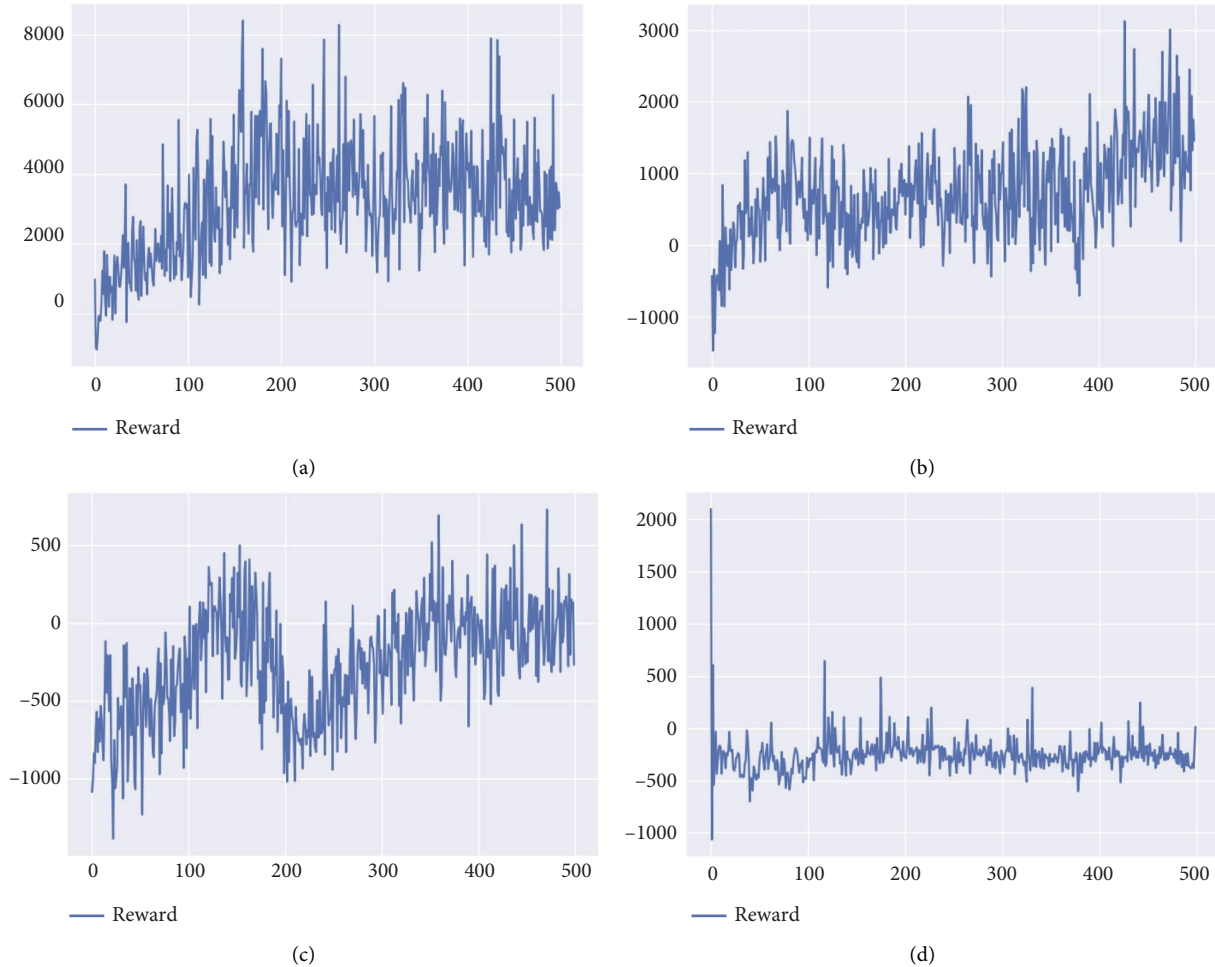


FIGURE 5: Different methods reward results: (a) IDDPG method, (b) DDPG method, (c) A2C method, and (d) DQN method.

sequences steps and divided by the total number of successfully executed attacks.

We added security rules from Table 2; there are now 3, 4, 5, and 6 security rules in total. In general, the longer the attack sequences are, even when the attacker cannot attack effectively, the more security rules there are. Additionally, Table 2 in Result A supports the opinion. As we can see, the attack sequence steps lengthen across all approaches as the number of security rules rises. Therefore, it is clear that the absence of security regulations in cyberspace makes it easier for attackers to capture their target, demonstrating the significance of the configuration of the cyberspace. The attack sequence steps may be used to gauge the degree of cyberspace security, as demonstrated by Result A. That is, if an assault succeeds more frequently, there is a lack of cyberspace security, which shows that the setup of the cyberspace is poor.

On the other hand, a longer attack sequence step indicates a better setup of the cyberspace. As a result, in our experiment, we can describe cyberspace security using the attack's sequence steps.

Following the experiment, we increased the number of security rules to 3 in order to confirm the effectiveness and value of our proposed method.

The reward received by the attacker is Result B.

The Results B are shown in Figure 5. First of all, as the quantity of training increases, R shows a gradual ascending trend until ultimately tending to converge. It is in the RL learning process, demonstrating that the proposed model complied with the RL features of discovering shorter hidden attack paths in cyberspace.

Second, Figure 5 shows four ways to get the award. We can see that, among the other methods, our proposed

TABLE 3: Attack successfully number and minimum steps results.

	Successfully number	Minimum steps
DQN	1203	1800
A2C	2533	2400
DDPG	3711	2100
IDDPG	6705	750

method, IDDPG, has the largest reward. This might explain the fact that our proposed method has shorter attack sequence steps and more effective attacks.

The attacker successfully attacking in an experiment is how we define the attack successfully number. The Minimum Steps is an episode's smallest successful attack sequence steps.

Result C is the attack successfully number.

Result D refers to the minimum steps.

Results C and D are recorded in Table 3.

Thirdly, we can see from Table 3 that our proposed method has the fewest attack sequence steps. It is also demonstrated that our proposed method is capable of discovering the attack sequences with the fewest steps under the same circumstances. The number of successful attacks is likewise at its greatest in the interim. Our proposed method is preferable to existing methods.

We may utilize the attack sequence steps as a gauge of the cyberspace security based on the experimental findings, which first establish that the configuration of the cyberspace can impact its security. Second, we employ a different method inside the same setting. Then we compare the reward, the attack successfully number, the minimum steps, and the attack successfully number using various methods in the same cyberspace environment. The experiment's highest reward went to our proposed method, which also had the most successful attacks and required the fewest steps. Our proposed method may attack more successfully and earn more rewards in an episode, according to the trial findings. As a consequence, the attacker will do the fewest steps, identify the cyberspace attack sequences that take the fewest steps. In conclusion, our proposed method is better than current methods.

5. Conclusion

We proposed a learning-based method to identify the cyberspace's vulnerability by locating the hidden attack path. In the meanwhile, we discovered the cyberspace vulnerability metrics, which are the steps an attacker successfully follows to launch an assault, and we ultimately conducted an experimental verification. This method has a great practical value because it can take into account the mutual effect of the multiple domain setup in cyberspace and use an intelligent way to analyze its weaknesses.

The RL method, which has shown improved results, is used in this research to analyze a typical cyberspace environment and discover hidden attack paths in configured cyberspace. The virtual environment in this work is constrained, though. We intend to use the RL to better manage

the operation and maintenance of additional cyberspace systems in the upcoming phase.

Data Availability

The data used to support the findings of this study can be obtained from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was results of the research project funded by the National Natural Science Foundation of China (No. 62076251 and 62106281). Our preprint manuscript has been submitted to arXiv (no. arXiv: 2104.07195) [32]. The authors shared the proposal on arXiv.

References

- [1] R. Rajkumar, I. Lee, L. Sha, and A. John, "Stankovic. "cyber-physical systems: the next computing revolution"" in *Proceedings of the CONFERENCE of the 47th Design Automation Conference, DAC 2010*, pp. 731–736, Anaheim, California, USA, July 2010.
- [2] D. D. Lee and M. Sugiyama, "Ulrike von Luxburg, isabelle guyon and roman garnett," in *Proceedings of the Advances in neural information processing systems 29: Annual conference on neural information processing systems*, pp. 5–10, barcelona, spain, december 2016.
- [3] N. Heo and P. K. Varshney, "Energy-efficient deployment of intelligent mobile sensor networks," *IEEE Trans. Systems, Man, and Cybernetics, Part A*, vol. 35, no. 1, pp. 78–92, 2005.
- [4] H. Yao, C. Jiang, and Yi Qian, "Intelligent Network Awareness," *Developing Networks using Artificial Intelligence*, Springer, Heidelberg, Germany, 2019.
- [5] S. S. Kim and A. L. Narasimha Reddy, "Statistical techniques for detecting traffic anomalies through packet header data," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 562–575, 2008.
- [6] A. Badea, V. Croitoru, and D. Gheorghicã, "Computer Networks Security Based on the Detection of User's Behavior," in *Proceedings of the 2015 9th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*, Bucharest, Romania, May 2015.
- [7] J. Dai, X. Sun, and P. Liu, "Patrol: revealing zero-day attack paths through network-wide system object dependencies," in *Jason Crampton, Sushil Jajodia and Keith Mayes, editors, Computer Security – ESORICS 2013*, pp. 536–555, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [8] T. Gonda, R. Puzis, and B. Shapira, "Scalable attack path finding for increased security," in *Shlomi Dolev and Sachin Lodha, editors, Cyber Security Cryptography and Machine Learning*, pp. 234–249, Springer International Publishing, Berlin, Heidelberg, 2017.
- [9] U. Garg, S. Geeta, K. Lalit, and A Awasthi, "Empirical analysis of attack graphs for mitigating critical paths and vulnerabilities," *Computers & Security*, vol. 77, pp. 349–359, 2018.
- [10] C. Basile, D. Canavese, L. Regano, P. Falcarin, and B. De Sutter, "A meta-model for software protections and

- reverse engineering attacks,” *Journal of Systems and Software*, vol. 150, no. 3–21, 2019.
- [11] A. Beutel, L. Akoglu, and C. Faloutsos, “Graph-based User Behavior Modeling: From Prediction to Fraud Detection,” in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Sydney, NSW, Australia, August 2015.
- [12] H. Kawazu, F. Toriumi, M. Takano, K. Wada, and I. Fukuda, “Analytical method of web user behavior using hidden Markov model,” in *Proceedings of the 2016 IEEE International Conference on Big Data (Big Data)*, Washington, DC, USA, December 2016.
- [13] H.-J. Liao, C. Hung Richard Lin, Y.-C. Lin, and K.-Y. Tung, “Intrusion detection system: a comprehensive review,” *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.
- [14] Ni Gao, L. Gao, H. E. Yi-Yue, and H. Wang, “A lightweight intrusion detection model based on autoencoder network with feature reduction,” *Acta Electronica Sinica*, vol. 45, 2017.
- [15] B. Xu, S. Chen, H. Zhang, and T. Wu, “Incremental k-nn svm method in intrusion detection,” in *Proceedings of the 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, China, November 2017.
- [16] B. Selvakumar and K. Muneeswaran, “Firefly algorithm based feature selection for network intrusion detection,” *Computers & Security*, vol. 81, pp. 148–155, 2019.
- [17] J. Zhang, M. Zulkernine, and A. Haque, “Random-forests-based network intrusion detection systems,” *IEEE Trans. Systems, Man, and Cybernetics, Part C*, vol. 38, no. 5, pp. 649–659, 2008.
- [18] I. M. Akashdeep and N. Kumar, “A feature reduced intrusion detection system using ANN classifier,” *Expert Systems with Applications*, vol. 88, pp. 249–257, 2017.
- [19] F. Qu, J. Zhang, Z. Shao, and S. Qi, “An intrusion detection model based on deep belief network,” in *Proceedings of the CONFERENCE of the VI International Conference on Network, Communication and Computing, ICNCC 2017*, pp. 97–101, Kunming, China, December 2017.
- [20] S. Nathan, T. Nguyen Ngoc, P. Vu Dinh, and S. Qi, “A deep learning approach to network intrusion detection,” *IEEE Trans. Emerging Topics in Comput. Intellig.* vol. 2, no. 1, pp. 41–50, 2018.
- [21] C. Yin, Y. Zhu, J. Fei, and X.-Z. He, “A deep learning approach for intrusion detection using recurrent neural networks,” *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
- [22] T. Thu Huong Le, J. Kim, and H. Kim, “An effective intrusion detection classifier using long short-term memory with gradient descent optimization,” in *Proceedings of the IEEE 2017 International Conference on Platform Technology and Service (PlatCon)*, pp. 1–6, Busan, South Korea, February 2017.
- [23] S. Naseer, Y. Saleem, S. Khalid, and M. K. Bashir, “Enhanced network anomaly detection based on deep neural networks,” *IEEE Access*, vol. 6, pp. 48231–48246, 2018.
- [24] R. S. Sutton and A. G. Barto, “Reinforcement learning: an introduction,” *IEEE Transactions on Neural Networks*, vol. 9, no. 5, p. 1054, 1998.
- [25] Y. Zhang, I. Clavera, B. Tsai, and P. Abbeel, “Asynchronous Methods for Model-Based Reinforcement Learning,” 2019, <https://arxiv.org/abs/1910.12453>.
- [26] T. P. Lillicrap, J. J. Hunt, P. Alexander, and H. Nicolas, “Continuous control with deep reinforcement learning,” 2015, <https://arxiv.org/abs/1509.02971>.
- [27] D. Silver, L. Guy, N. Heess, and T. Degris, “Deterministic policy gradient algorithms,” *31st International Conference on Machine Learning, ICML*, vol. 1, p. 06, 2014.
- [28] Q. Zhang, K. Dehghanpour, Z. Wang, and Q. Huang, “A learning-based power management method for networked microgrids under incomplete information,” *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1193–1204, 2020.
- [29] Q. Zhang, K. Dehghanpour, Z. Wang, F. Qiu, and D. Zhao, “Multi-agent safe policy learning for power management of networked microgrids,” *IEEE Transactions on Smart Grid*, vol. 12, no. 2, pp. 1048–1062, 2021.
- [30] R. Elderman, L. Pater, T. Albert, M. Drugan, and W. Marco, “Adversarial reinforcement learning in a cyber security simulation,” in *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART), 2017. 9th International Conference on Agents and Artificial Intelligent Conference Date: 24-02-2017 through 26-02-2017*, Porto, Portugal, February 2017.
- [31] S. Sharma, M. Ishizawa, D. Chan, and D. Lavoué, “16-year simulation of arctic black carbon: transport, source contribution, and sensitivity analysis on deposition,” *Journal of Geophysical Research: Atmospheres*, vol. 118, no. 2, pp. 943–964, 2013.
- [32] L. Zhang, W. Bai, W. Li, S. Xia, and Q. Zheng, “Discover the hidden attack path in multi-domain cyberspace based on reinforcement learning,” 2021, <https://arxiv.org/abs/2104.07195>.