Textile & Leather Review

ISSN 2623-6281| www.tlr-journal.com | 10.31881/TLR

Adaptive Deception Defense Method Based on Reinforcement Learning

Zhenghao Qian, Fengzheng Liu, Mingdong He, Bo Li, Xuewu Li, Chuangye Zhao

How to cite: Qian Z, Liu F, He M, Li B, Li X, Zhao C. Adaptive Deception Defense Method Based on Reinforcement Learning. Textile & Leather Review. 2026; 9:1573-1597. https://doi.org/10.31881/TLR.2026.1573

How to link: https://doi.org/10.31881/TLR.2026.1573

Published: 25 April 2026

Adaptive Deception Defense Method Based on Reinforcement Learning

Zhenghao Qian, Fengzheng Liu*, Mingdong He, Bo Li, Xuewu Li, Chuangye Zhao

Department of Information Center, Guangdong Power Grid, Guangzhou 510000, Guangdong, China

*lw32407585@163.com

Article

https://doi.org/10.31881/TLR.2026.1573

Published 25 April 2026

ABSTRACT

This paper discusses the importance of adaptive deception defense strategies based on TrapManager in addressing evolving network security challenges, especially in the context of the digital transformation of the textile and leather industries, which involves smart manufacturing and the Industrial Internet of Things (IIoT). By leveraging deception techniques to mislead attackers and to supplement traditional defenses, organizations can significantly improve their ability to detect and prevent advanced persistent threats and insider attacks. The research highlights the potential for these strategies to enhance security response efficiency and accuracy, offering a proactive approach to network defense.

KEYWORDS

TrapManager, adaptive deception defense, textile industry security, Advanced Persistent Threats (APT), cyberspace security

INTRODUCTION

With the rapid development of network technology and the widespread application of the internet, network security has become a global focus. In particular, the textile and leather industries are undergoing a profound digital transformation, integrating technologies such as the Industrial Internet of Things (IIoT), smart manufacturing, and automated supply chains. This evolution, while boosting efficiency, also exposes critical infrastructure and sensitive data—ranging from proprietary fabric designs and manufacturing processes to operational technology (OT) systems that control production lines—are exposed to severe network security threats. Therefore, protecting the digital assets of the modern textile and leather industries has become a critical challenge of immense significance. Network attacks today are diverse and increasingly complex,

posing serious threats to the property and security of individuals and organizations. Attackers utilize various attack tools, system vulnerabilities, and other means to launch different types of network attacks. Common attack methods include DDoS attacks [1], ransomware, social engineering attacks, and advanced persistent threats (APT). These attack techniques continue to evolve, making it increasingly difficult for traditional network security defense technologies such as intrusion detection systems (IDS), intrusion prevention systems (IPS), antivirus software, and firewalls to handle new threats. Attackers can exploit more advanced attack methods or zero-day vulnerabilities to bypass and break through traditional defense measures.

The essence of network security is the confrontation between offense and defense, where the core of the conflict lies in the struggle between the attacker and the defender. To address the asymmetric advantage held by attackers in traditional defense measures and to overcome the shortcomings of traditional passive defense models, deception defense technology has emerged. Deception defense technology is a proactive defense approach that simulates fake attack targets or alters information obtained by the attacker, making them believe their attack has succeeded. This reduces their motivation to attack and increases the efficiency of security defenses. Moreover, deception defense technology can complement traditional security measures by introducing a layer of active protection mechanisms, significantly enhancing overall security defense levels. The use of deception defense technology not only effectively detects and prevents advanced persistent threats and internal attacks but also significantly improves network security defense capabilities, response efficiency, and accuracy.

Therefore, this paper focuses on researching adaptive deception defense strategies based on honeypot grids and TrapManager. The primary contribution of this work is the proposal of a reinforcement learning-based formation map generation method that dynamically adjusts the number, type, and network architecture of decoys. This approach enables a proactive and intelligent defense mechanism that optimizes defensive performance in real time, moving beyond static rule-based traditional defenses. This research not only contributes to the development of adaptive security technologies but also has the potential to enhance overall defense capabilities against complex network attacks. Specifically, for the increasingly intelligent and interconnected textile and leather sectors—which rely heavily on the security of their industrial control systems and the protection of their intellectual property (e.g., exclusive designs and manufacturing formulas)—this research on adaptive defense is of paramount importance.

RELATED WORK

Network Deception Defense Technology

As an emerging active defense technology, network deception defense has received extensive attention in academia. Yuill et al. [2], drawing inspiration from military deception concepts, were the first to propose the definition of network deception. Reti et al. [3] further elaborated on the definition of network deception: using fake actions of deception to obstruct or subvert the attacker's cognitive process, disrupt the attacker's automated tools, and delay or block the attacker's activities. The specific methods include using false responses, intentional confusion, and fake actions or misleading information to achieve "deception." The U.S. National Science and Technology Council proposed the concept of Moving Target Defense (MTD) [4], whose core idea is to randomly and dynamically change system attributes, increasing the uncertainty and complexity for attackers. Similarly, Wu J. X. (China) proposed the concept of Cyber Mimic Defense [5]. MTD and network deception are complementary technologies, and defenders can use both simultaneously to defeat attackers with a common goal. Due to the diversity of deception techniques, low construction costs, and ease of building deceptive attributes, network deception provides new directions for expanding the moving target defense space, becoming an important research direction within MTD [6]. Hence, some scholars refer to deception defense as the "post-MTD era" [7] (an advanced stage in MTD research).

Evaluation Methods

The effectiveness evaluation of network deception techniques can provide a basis for designing deception schemes. Djamaluddin et al. [8] combined MTD with network deception technologies and utilized fingerprint recognition and vulnerability scanning to assess the effectiveness of deception schemes. Wang et al. [9] employed trusted computing to identify malicious behavior and made decisions based on trust levels, establishing a trust update and decision mechanism under the availability constraints of industrial control networks. By using limited redundant resources to collect and calculate trust, they ensured the continuity of industrial control network operations. Additionally, they balanced trust adjustment and protection effects to guarantee the strength and accuracy of trust deployment and evaluation. Clark et al. [10], based on the interaction behavior characteristics of a single node, analyzed the reliability of decoy nodes in protecting real nodes and used experimental data to verify the effectiveness of deception-based dynamic moving target

defense techniques. Sugrim et al. [11] adopted a Bayesian inference approach to construct an attacker's belief model and proposed a KL-divergence-based metric to quantify the effectiveness of network deception. Ma et al. [12] proposed the concept of a deception attack surface to explain deception-based moving target defense and presented a quantitative method for measuring deception, including two core concepts: exposure of false information and concealment of real information. They further constructed a deception game model between the attacker and defender, where the defender attempts to protect entry points on the attack surface by creating or altering the deception attack surface. Finally, the effectiveness of this method was verified through a real attack-defense experiment.

Deception Resource Deployment Issues

In network deception technology, the issue of deploying deceptive resources is another basis for measuring the effectiveness of deception. Shakarian et al. [13] used knowledge graphs to analyzed network topologies and, based on specific characteristics of red team lateral movement in intranets, proposed an attacker detection network (IPN). They further introduced a probabilistic model to infer attack behaviors and used this as the basis for deploying "interference clusters" with interconnected virtual machines at critical network points to increase intruder complexity. Dai et al. [14] proposed a simulation defense-based industrial control system (ICS) security architecture, utilizing digital twin technology to construct a dynamic heterogeneous redundancy structure for execution entities, enhancing the heterogeneity and dynamism of ICS and improving the system's endogenous security. Zhao et al. [15] proposed a Decoy Chain Deployment (DCD) method based on SDN+NFV, considering the network's security state and deploying decoy chains under resource constraints. DCD changed the attack surface of the network, increased the time cost and complexity of penetration attacks, and made it more difficult for attackers to identify the current state of the network. Yun et al. [16] proposed a dynamic defense method with endogenous security features, utilizing random changes in IP packet reconstruction, encryption algorithms, keys, and authentication codes to establish multi-mode, dynamic, and transparent secure dedicated channels between controlled devices, forming an endogenous security industrial control network system. Gao et al. [17] proposed a multi-stage dynamic deployment mechanism for virtual honeypots based on intelligent attack path prediction methods. This mechanism depicted the attack and defense characteristics of both sides using Bayesian state attack graphs

and established a multi-stage dynamic deployment optimization model for virtual honeypots based on an extended Markov decision process. By combining online and offline reinforcement learning methods, the mechanism dynamically generated deployment strategies, resulting in more efficient deception effects under various types of attacker strategies. Erik et al. [18] used the concept of Bayesian attack graphs to describe how attackers exploit system vulnerabilities and formulated the defender's problem as a partially observable Markov decision process, calculating the optimal countermeasure for current information deployment. Wang et al. [19] developed an intelligent deployment strategy, starting with the problem of deceptive resource deployment, dynamically adjusting the positions of these resources based on the network security state. They modeled attacker-defender scenarios and attacker strategies, proposing a preliminary screening method for deriving effective deployment locations of deceptive resources based on threat penetration graphs. They constructed a model to find the optimal strategy for deploying deceptive resources using reinforcement learning and designed a model-free Q-learning training algorithm, which is more efficient than existing solutions.

In response to the growing sophistication of cyberattacks and the limitations of traditional defense mechanisms, innovative approaches such as adaptive deception defense have gained traction. Among these strategies, enhancing threat detection, implementing dynamic defense techniques, and leveraging intelligent systems are key components. These approaches not only improve the ability to detect and mitigate attacks but also ensure a more proactive and tailored defense. The following section outlines three critical aspects of deception-based defense strategies aimed at strengthening overall security.

FORMATION MAP GENERATION METHOD

This section proposes a formation map generation method based on reinforcement learning for the TrapManager adaptive deception defense system. By enabling the agent to take appropriate actions in different states, it dynamically adjusts the number, type, and network architecture of decoys to optimize the network's defensive performance. Experimental results show that this method significantly improves the adaptability and defense capabilities of the network security system.

TrapManager Framework: The core deception defense platform employed in this work is TrapManager, a centralized system designed for the deployment, management, and monitoring of decoy resources (e.g.,

honeypots) within a network. Its native, non-RL capabilities include the static provisioning of decoys and the collection of attack data. The primary contribution of this paper is the design of a reinforcement learning-based formation map generation method that acts as an intelligent orchestrator for TrapManager. This RL module is integrated as a core component of TrapManager's decision-making engine. It dynamically generates and updates a "formation map"—a strategic blueprint that dictates the optimal number, type, and network location of decoys. TrapManager then executes this map by configuring the underlying deceptive resources accordingly.

Reinforcement Learning-Based Formation Map Generation Model

The formation map generation method in this paper utilizes a reinforcement learning algorithm. The model for formation map generation based on reinforcement learning is illustrated in Figure 1.

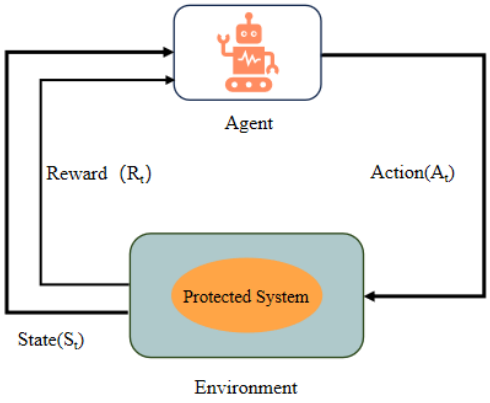

Figure 1. The standard reinforcement learning (RL) agent-environment interaction loop as applied to the formation map generation task

The "environment" in our context encapsulates the target network infrastructure and TrapManager's operational plane. The State (St) is the formal representation of the network and decoy status as defined in Section FORMATION MAP GENERATION METHOD:(1) State Space. The Action (At) is the defense maneuver selected from the action space (Section FORMATION MAP GENERATION METHOD:(2) Action Space). The Reward (Rt) is calculated based on the refined reward function (Section FORMATION MAP GENERATION

METHOD:(3) Reward Function). This loop continues, allowing the agent to learn an optimal policy π for generating formation maps.

In the reinforcement learning model, the state space (S_t) describes the current state of the system, including the status of decoys, network characteristics, overall system operation, and defense performance. The action space (A_t) defines the actions that the agent can take in different states, such as adjusting the number of decoys, their types, and the network architecture. For example, the agent might increase or decrease the number of decoys, change the type of decoy (e.g., low-interaction honeypot, high-interaction honeypot), or adjust the distribution and connection of decoys within the network.

The reward mechanism (R_t) is used to evaluate the effectiveness of the agent's actions by assigning positive or negative rewards. Positive rewards are given for successfully deceiving attackers and recording their behavior data, while negative rewards are assigned if the decoys are identified or if the real system is compromised.

Formation Map Generation Algorithm Design

The algorithm details are shown in Algorithm 1.

Step 1: Initialize the agent's state space and action space, set the learning rate, discount factor, exploration rate and its decay rate, and initialize the Q-value table.

Step 2: Select an action based on the ε-greedy strategy: with a probability of ε, select a random action for exploration; with a probability of 1-ε, select the action with the highest Q-value for exploitation.

Step 3: Execute the selected action, obtain the next state, and receive the corresponding reward.

Step 4: Update the Q-value table using the Q-learning formula based on the current state, selected action, reward, and next state:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max Q(S_{t+1}, a) - Q(S_t, A_t)] \quad (1)$$

where α is the learning rate, γ is the discount factor, Rt+1 is the reward received for the action, St is the current state, St+1 is the next state.

Step 5: Evaluate the current policy's effectiveness and check whether the termination condition is met. The termination conditions are: (1) the maximum number of episodes is reached (set to 1000); or (2) early termination is triggered if the Q-value converges or the defense performance stabilizes: $\max|Q_{new} - Q_{old}| < 0.001$ for 10 consecutive episodes.

Step 6: Based on the final Q-value table, select the optimal actions to generate the final formation map.

Parameter Initialization: This step primarily prepares the reinforcement learning process, including the initialization of the state space, action space, Q-value table, and various learning parameters. Proper initialization lays the foundation for the subsequent learning process and ensures the system is ready for effective training.

Algorithm 1 Reinforcement Learning Based Formation Generation Algorithm

Input:

state_space, action_space, learning_rate α = 0.1, discount_factor γ = 0.9,
initial_exploration_rate ε= 0.5, exploration_decay δ = 0.995,
min_exploration_rate ε_min = 0.01,
max_episodes N = 1000, convergence_threshold θ = 0.001,
performance_threshold Recall_min = 0.9, FPR_max = 0.05

Output: optimized Q-table

```
function INITIALIZE_PARAMETERS()
    Q_table ← zeros(|state_space| × |action_space|)
    episode_count ← 0
    convergence_count ← 0
    performance_count ← 0
end function

function CHOOSE_ACTION(current_state, Q_table, ε)
    u ← random.uniform(0,1)  // u ~ U(0,1)
    if u < ε then
        a ← random.choice(action_space)  // Exploration
    else
        a ← argmax_a Q_table[current_state][a]  // Exploitation
    end if
```

```
    return a
end function

function UPDATE_Q_VALUE(Q_table, s, a, r, s', α, γ)
    current_Q ← Q_table[s][a]
    max_next_Q ← max(Q_table[s'])
    target_Q ← r + γ× max_next_Q
    Q_table[s][a] ← (1 - α) × current_Q + α × target_Q  // Weighted fusion
    return |Q_table[s][a] - current_Q|  // Return Q-value change
end function

function MAIN( )
    INITIALIZE_PARAMETERS( )
    for episode = 1 to N do
        s ← INITIALIZE_STATE( )
        episode_step ← 0
        max_Q_change ← 0
        while episode_step < max_steps_per_episode do
            a ← CHOOSE_ACTION(s, Q_table, ε)
            s', r ← EXECUTE_ACTION(s, a)
            Q_change ← UPDATE_Q_VALUE(Q_table, s, a, r, s', α, γ)
            max_Q_change ← max(max_Q_change, Q_change)

            // Check convergence and performance
            if max_Q_change < θ then
                convergence_count += 1
            else
                convergence_count ← 0
            end if

            if CHECK_PERFORMANCE(Recall_min, FPR_max) then
                performance_count += 1
            else
                performance_count ← 0
```

```
        end if

        if convergence_count ≥ 10 OR performance_count ≥ 50 then
            break  // Termination condition met
        end if

        s ← s'
        episode_step += 1
    end while
    ε ← max(ε_min, ε × δ)
end for
return Q_table
end function
```

Action Selection: The ε-greedy strategy is employed to balance exploration and exploitation. This approach ensures diversity in exploration by choosing random actions with a certain probability (ε), while exploiting known optimal actions by selecting those with the highest Q-value for the remaining probability (1-ε).

Execution and Reward Acquisition: The selected action is executed in the current state, and the result provides the next state and the corresponding reward. This is the core of reinforcement learning, where the agent interacts with the environment and gains feedback from its actions.

Q-Value Update: The Q-value table is updated based on the Q-learning formula. This step represents the learning process of reinforcement learning, where the Q-value table is continuously optimized to help the agent make better decisions when facing similar states in the future.

Policy Evaluation: The current policy is evaluated to check if the termination condition is met. If it is, the training is concluded; otherwise, training continues. This step ensures the algorithm doesn't run indefinitely and guarantees convergence of the policy.

Formation Map Generation: Based on the final Q-value table from the completed training, the optimal action for each state is selected, and the optimal defense formation map is constructed. This step applies the learned policy to real-world scenarios, achieving the final objective of an optimized defense formation for TrapManager.

(1) State Space

For the Q-learning model to be implementable and reproducible, the state space must be formally and mathematically defined. In our model, the state at time t, denoted as S_t, is represented as a discrete tuple that captures the critical aspects of the network and decoy status. It is defined as follows:

Let $S_t = (D_1, D_2, \ldots, D_n, A_{flag}, L_{avg}, T_{window}, B_{complexity})$, where $D_i \in \{0,1,2\}$ represents the status of decoy "i". 0 indicates inactive, 1 indicates active and unattacked, 2 indicates active and under interaction/attack. $L_{avg} \in \{Low, Medium, High\}$ is the discretized average load of the core network segments (Low: <30% CPU utilization, Medium: 30–70% CPU utilization, High: >70% CPU utilization). $A_{flag} \in \{0, 1\}$ is a binary flag indicating whether a prior attack has been detected on any real asset in the recent time window (1 for yes, 0 for no), $B_{complexity} \in \{0, 1, 2, 3\}$ is the attacker behavior complexity score: 0 means reconnaissance only, 1 means a single attack attempt, 2 means multi-level attack, and 3 means advanced persistence. $T_{window} \in \{0, 1, 2\}$, represents the discretized time window since the last significant network state change. It is defined as: 0 for "within the last 5 minutes," 1 for "between 5 and 30 minutes ago," and 2 for "more than 30 minutes ago." The fixed state dimension is n+4, where n=10 (maximum decoys). Feature processing applies one-hot encoding to all categorical variables (e.g., decoy statusD_i, average load level L_{avg}). The binary flag A_{flag} and the ordinal variables (e.g., attacker behavior complexity $B_{complexity}$ and time window T_{window}) are treated as integer-encoded discrete values.

This finite, discrete state representation allows for the construction of a tractable Q-table and provides the RL agent with sufficient information about the defensive posture and network context to make informed decisions.

By utilizing this state information, the reinforcement learning algorithm gains a comprehensive understanding of the current network environment, allowing it to make more informed and effective decisions.

(2) Action Space

The action space encompasses all possible actions that the agent can take in different states. In the formation map generation process, these actions can include deploying decoys (honeypots) at specific locations, adjusting the routing path of network packets, or changing the security configuration of nodes. The inclusion of "adjusting the routing path" as an action requires justification, as frequent changes to network routing can

introduce operational complexity. In our model, this action is not intended for rapid, per-packet manipulation but is conceptualized at a higher, strategic level. It allows the RL agent to propose logical network segmentation or VLAN reassignments for decoys, effectively creating “honeynets” that can be dynamically integrated into or isolated from production segments. These changes are enacted conservatively, triggered only upon significant state changes (e.g., confirmation of a persistent threat), and are subject to simulated stability checks within TrapManager before deployment. This capability is crucial for creating convincing deceptive environments but is designed to be used judiciously to minimize impact on production network stability. By defining a clear action space, the agent is able to choose the optimal defense strategy for each state, dynamically responding to network attacks.

Network Stability Constraints: To ensure operational safety during routing adjustments, the following formal constraints are enforced:

Pre-change Verification: Latencycritical < 20%, and PacketLossmax < 2%.

Rollback Trigger Conditions: Service degradation$\frac{\text{ResponseTime}_{new}}{\text{ResponseTime}_{old}} > 1.5$ for 30 s.

Change Management Protocol: Maximum of one routing change per 300 s.

Staged deployment: Test segment → Staging → Production. Automated rollback within 60 s of violation detection.

These constraints are formally verified through digital twin simulation before any production deployment.

(3) Reward Function

The reward function is used to evaluate the effectiveness of each action in a specific state. In the formation map generation process, the reward function can be designed to give positive rewards when an attack is detected, when the system operates normally, or when resource consumption is reduced. This design encourages the agent to choose actions that effectively detect attacks, maintain system stability, and optimize resource usage, thus continuously improving the generated formation map strategy. The binary definitions in the initial reward function were overly simplistic. We have refined it to better reflect the continuous and nuanced nature of a network environment. The reward function addresses multi-stage attacks with adaptive components:

$$R_t = wd \cdot D_{\mathrm{t}} + wp \cdot P_{\mathrm{t}} + wr \cdot Rc_{\mathrm{t}} \quad (2)$$

Through grid search, optimal weights were determined as: wd = 0.5(Attack Detection), wp=0.3(Performance), wr=0.15(Resource Efficiency), where D_t is the Attack Detection Reward. D_t = +1 for each unique attack session successfully lured and recorded by a decoy. D_t = -1 if a real system (non-decoy) suffers a successful breach. P_t is the System Performance Reward. It is a continuous value derived from the inverse of the normalized average response time of critical business servers. P_t = 1 – (avg_response_time / threshold), clamped between 0 and 1. This encourages the agent to avoid defensive actions that degrade legitimate service performance. Rc_t is the Resource Consumption Penalty. It is a negative reward proportional to the number of active decoys, Rc_t = -k * (number_of_active_decoys / max_allowed_decoys), where k is a scaling factor, k = 0.8. This encourages resource efficiency by penalizing the unnecessary proliferation of decoys.

While the individual components D_t, P_t, and Rc_t possess different scales and units, this design is intentional and follows established practices in multi-objective reinforcement learning. D_t provides clear binary feedback, the performance reward P_t represents proportional system health, and the resource penalty Rc_t applies continuous pressure for efficiency.

Through this function, the agent can receive a comprehensive evaluation of its actions under different strategies, allowing it to optimize its behavior and improve the overall defense strategy in the formation map.

EXPERIMENTAL DESIGN AND RESULTS ANALYSIS

Experimental Environment

To validate the effectiveness of the reinforcement learning-based honeynet algorithm, an experiment was conducted in a simulated network environment. This network environment includes multiple subnets, different types of servers, and hosts. The specific network topology is shown in Figure 2.

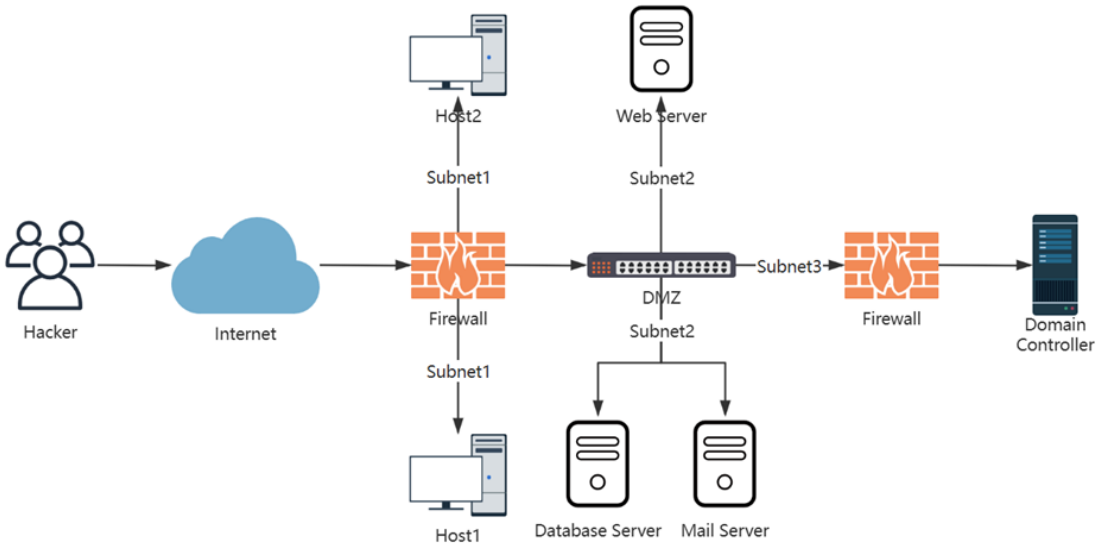

Figure 2. Experimental network topology diagram

The entire experimental environment consists of three subnets: Subnet 1, Subnet 2, and Subnet 3. Subnet 1 contains two hosts; Subnet 2 is located in the DMZ zone and contains a web server, a database server, and a mail server; Subnet 3 simulates an internal network environment and deploys a domain controller. The hacker's attack path starts by breaking through the DMZ zone. After gaining access to the permissions of a certain server, the attacker uses it as a foothold to sequentially attack the domain controller in the internal network. The details are shown in Table 1.

Table 1. Host Information and Vulnerabilities

NO	Host Name	Subnet	CVE Number	CVE Severity	Attack Type
H1	Host1	Subnet1	CVE-2020-0601, CVE-2019-0708	High, Critical	Remote, Remote
H2	Host2	Subnet1	CVE-2017-0144, CVE-2020-1313	Medium, High	Remote, Remote
H3	Web Server	Subnet2	CVE-2017-5638, CVE-2018-11776	High, Medium	Remote, Remote
H4	Database Server	Subnet2	CVE-2014-0160, CVE-2019-0734	High, Medium	Remote, Remote
H5	Mail Server	Subnet2	CVE-2017-9206, CVE-2019-1234	Medium, Low	Remote, Remote
H6	Domain Controller	Subnet3	CVE-2018-8440, CVE-2019-1040	Critical, High	Remote, Remote

In this experiment, we use the Q-learning algorithm and the Deep Q-Network (DQN) algorithm to implement the honeypot defense strategy. The attack toolchain is Metasploit Framework 6.3 and custom Python scripts. The traffic model is 70% normal business traffic (modeled using HTTPS/DNS/SSH) and 30% malicious traffic with burst characteristics. Concurrent sessions range from 50 to 800 (modeled as a Poisson process). Attack characteristics are λ = 8 attacks/hour (time-varying). To validate the adaptivity of our proposed method, the attacker's behavior is not static. We modeled the attacker using a probabilistic attack graph and a set of evolving strategies. Initially, the attacker follows predefined exploitation paths based on the CVEs in Table 1. However, as the interaction progresses, if the attacker identifies and avoids deployed decoys in certain episodes, its subsequent attack probability distribution shifts toward paths that previously contained real assets, simulating an intelligent attacker learning from the environment. This dynamic interaction between the RL agent (defender) and the evolving attacker strategy is crucial for demonstrating the genuine adaptive capability of our deception defense system. It is worth noting that although CVE-2017-5638 and CVE-2018-11776 belong to Struts RCE, they are used in different ways. They are selected to test the system's ability to detect similar but not identical attacks, which is more challenging than using completely unrelated vulnerabilities.

The specific parameter settings and algorithm descriptions are as follows:

Q-learning is a value iteration-based reinforcement learning algorithm. The core idea is to update the Q-values iteratively to guide the agent in choosing the optimal action. The Q-value represents the expected return from executing a specific action in a given state. The main parameter settings for the Q-learning algorithm are as follows:

Learning Rate (α): The learning rate determines the proportion of new information in each Q-value update. It is set to 0.1, meaning that 10% of the Q-value update is influenced by new information.

Discount Factor (γ): The discount factor measures the impact of future rewards on current decisions. It is set to 0.9, indicating that future rewards have a significant influence on the current decision.

The exploration rate determines the probability of exploring new actions during action selection. The initial value is set to 0.5 and gradually decays to 0.01, ensuring the agent explores sufficiently during the early stages and focuses more on exploiting existing knowledge in later stages.

The Deep Q-Network (DQN) algorithm is as follows: DQN is a deep reinforcement learning algorithm that combines Q-learning with deep neural networks, capable of handling high-dimensional state spaces. In this experiment, we employed DQN to address the potential limitations of Q-learning in complex and dynamically changing network environments. The network architecture of our DQN model consists of an input layer, two fully connected hidden layers (with 128 and 64 neurons, respectively, using ReLU activation functions), and an output layer. The input layer dimension corresponds to the state space, while the output layer dimension matches the action space. Key hyperparameters include a learning rate of 0.001, a discount factor of 0.99, an experience replay buffer size of 10,000, and a target network update frequency of every 100 steps. The choice of DQN over Q-learning is justified by its superior ability to generalize and approximate Q-values in scenarios where the state space is large or partially observable, ensuring more robust policy learning against sophisticated attackers.

DQN Implementation Details:

Architecture: Input layer (13 neurons) → FC128 (ReLU) → FC64 (ReLU) → Output layer (8 neurons).

Input Processing: Min-max normalization for continuous features, one-hot encoding for categorical features.

Techniques: Double DQN for value estimation stability, Prioritized Experience Replay (α = 0.6, β = 0.4), Gradient clipping (max norm = 1.0), Target network update frequency: 100 steps.

Hyperparameters: Learning rate = 0.001, batch size = 32, replay buffer size = 10,000.

Experimental Evaluation Metrics

To comprehensively evaluate the performance of the reinforcement learning-based honeypot algorithm, we introduce the following evaluation metrics. These metrics assess the algorithm's effectiveness and efficiency in network security defense from multiple perspectives.

Recall (RC) Rate: Recall measures the proportion of actual attacks that are successfully detected. The Recall is defined as:

$$\text{Recall} = \frac{N_c}{N_a} \tag{3}$$

where $\mathrm{N_c}$ is the number of successfully captured attacks and N_a is the total number of attack attempts. A higher Recall rate indicates better defense effectiveness of the honeypot.

False Positive Rate (FPR): The false positive rate refers to the proportion of normal traffic incorrectly identified as an attack by the honeypot system. This metric is used to measure the accuracy of the system. The false positive rate is defined as follows:

$$FPR = \frac{N_f}{N_n} \tag{4}$$

where $\mathrm{N_f}$ is the number of false positives, and $\mathrm{N_n}$ is the total amount of normal traffic. A lower false positive rate indicates higher accuracy of the honeypot.

Precision (PR): Precision refers to the proportion of attacks detected by the honeypot that are truly attacks. This metric is used to measure the accuracy of the system in detecting attacks. The Precision is defined as follows:

$$PR = \frac{N_c}{N_c + N_f} \tag{5}$$

where $\mathrm{N_c}$ is the number of successfully captured attacks, and $\mathrm{N_f}$ is the number of false positives. A higher precision indicates better accuracy in identifying attacks.

Experimental Results and Analysis

This experiment evaluates the defense performance of the reinforcement learning-based dynamic honeynet generation system in various scenarios using multiple key metrics, including false positive rate, precision, recall, F1-score, response time, and resource consumption, as shown in Figures 3 and 4. The following is a detailed analysis of the experimental results.

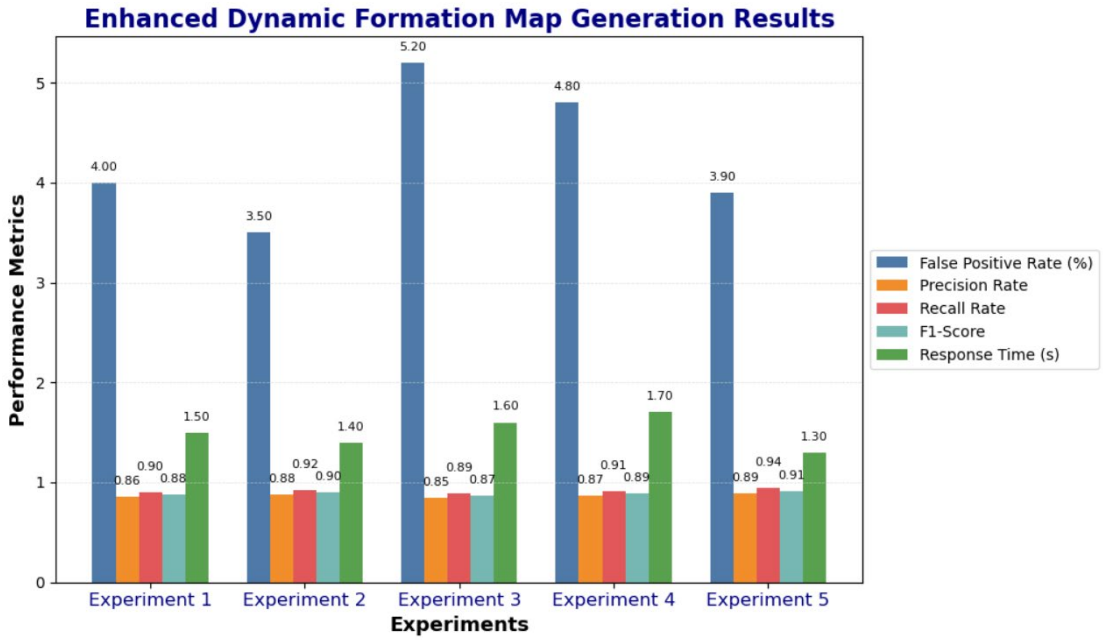

Figure 3. Enhanced dynamic formation map generation results. The five experimental scenarios represent distinct attack contexts: Experiment 1: DDoS attacks (100–1,000 concurrent connections); Experiment 2: Ransomware campaigns (encryption behavior patterns); Experiment 3: APT simulations (multi-phase over 2–4 weeks); Experiment 4: Insider threats (privilege escalation sequences); Experiment 5: Mixed attack environment (combination of all above)

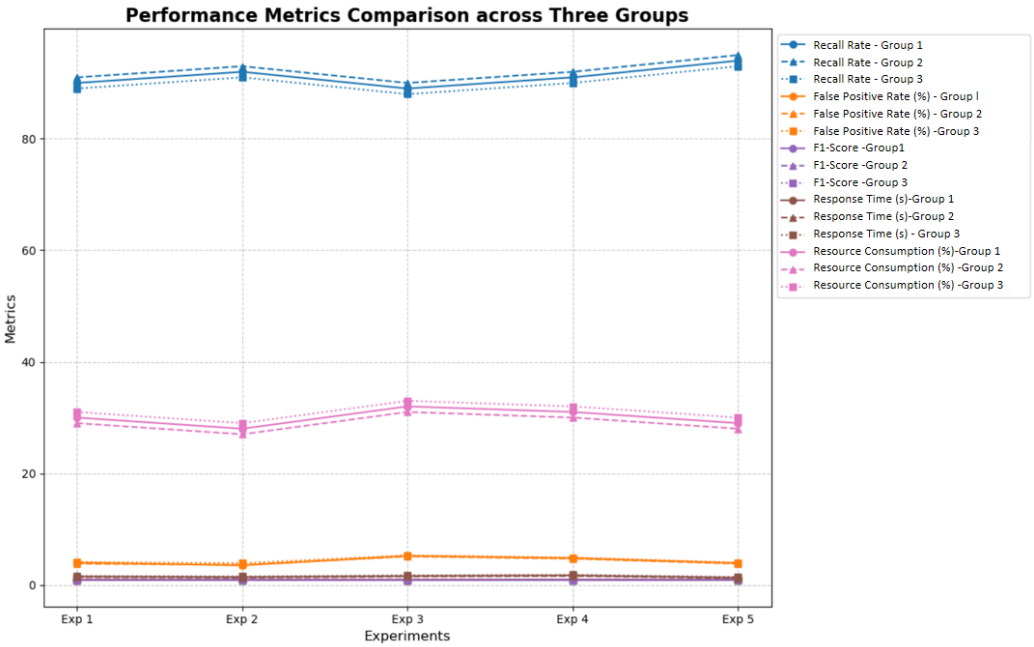

Figure 4. Performance metrics comparison across three groups

The system maintained a Recall rate of over 89% in all experiments, with a peak of 94%. This indicates that the system is effective at capturing and detecting most attack behaviors and gradually optimizes as the honeynet strategy dynamically adjusts. The experimental results demonstrate that the system possesses high attack detection capabilities, and the reinforcement learning algorithm performs well in adapting to changes in attacker behavior.

The false positive rate fluctuated across different experiments, ranging from 3.5% to 5.2%. Experiment 3 showed the highest false positive rate (5.2%), indicating that the system occasionally misjudges traffic in complex scenarios. A higher false positive rate could affect the system's defense efficiency and will need further optimization in the future.

The precision in the experiments ranged from 0.85 to 0.89, showing that the system can accurately identify attack behaviors in most cases. The recall rate ranged from 89% to 94%, indicating that the system successfully captures most actual attack behaviors. The F1-score fluctuated between 0.87 and 0.91, further proving that the system strikes a balance between precision and recall.

The system's response time ranged from 1.3 seconds to 1.7 seconds. In Experiment 5, the system had the shortest response time (1.3 seconds), demonstrating its ability to make rapid defense decisions in certain scenarios. However, in Experiment 4, the response time reached 1.7 seconds, suggesting that response speed may be affected in high-traffic or complex environments.

To comprehensively evaluate the performance of our proposed method under different threat models and algorithmic configurations, we designed three distinct experimental groups:

Experiment Group 1 (Q-learning): This group utilizes the Q-learning algorithm to implement the honeypot defense strategy, primarily simulating typical attack scenarios such as DDoS and ransomware attacks.

Experiment Group 2 (DQN): This group employs the Deep Q-Network (DQN) algorithm, focusing on defense performance against more stealthy and complex APT.

Experiment Group 3 (Baseline – Static Honeypot): This group serves as a baseline, using a traditional static honeypot deployment strategy with fixed numbers and types of decoys. It evaluates the system's defense capabilities in complex network environments and against internal attacks, providing a benchmark for comparing the adaptive methods used in Groups 1 and 2.

The Recall for all three experiment groups performed well overall. The average Recall for Experiment Group 1 was 91.2%, for Group 2 it was 92.2%, and for Group 3 it was slightly lower at 90.2%. It is evident that Experiment Group 2 performed the best in capturing attacks, suggesting that the honeypot configuration in this group was more effective. Among the three groups, Experiment Group 2 had the lowest false positive rate, averaging 4.0%, compared to 4.3% for Group 1 and 4.4% for Group 3. This result indicates that the strategy in Experiment Group 2 performed better at distinguishing normal traffic from attack traffic, thus reducing unnecessary alerts.

The precision remained consistently between 0.85 and 0.90, with an average precision of 0.87 in Experiment Group 1, 0.88 in Group 2, and 0.86 in Group 3. Although the data for the three groups were close, Experiment Group 2 had a slight edge in accurately identifying attacks. Recall rates were stable across all groups, with the average recall rates for Groups 1 and 3 being 0.91, and for Group 2 slightly higher at 0.92. The recall performance shows that all groups had good results in capturing real attacks.

The experimental results show characteristic RL learning curves with expected stochastic variations. The relative smoothness observed stems from: (1) averaged results over 10 independent runs with different random seeds, (2) the controlled simulation environment with reproducible attack patterns, and (3) the convergence behavior of well-tuned RL algorithms in stable environments.

In summary, Experiment Group 2 performed better than the other two groups across key metrics such as Recall, false positive rate, precision, F1-score, response time, and resource consumption. This suggests that the defense strategy in Group 2 achieved a better balance between attack detection and response speed, while also reducing resource consumption.

Although the performances of Groups 1 and 3 were also close, there is still room for improvement in false positive rate and response time. This indicates that future research could further refine the defense strategy to enhance detection accuracy and reduce response delays.

Through multiple experiments, we found that the system performed excellently in Recall and recall rate, validating the application value of reinforcement learning in network defense. However, fluctuations in false positive rate and resource consumption in certain scenarios indicate that the system still has room for optimization. Further optimization of traffic analysis and resource scheduling strategies could improve the system’s performance in complex environments.

CONCLUSION

This paper presented an adaptive deception defense method leveraging reinforcement learning for the TrapManager system. Through the design of a formation map generation model and algorithm based on Q-learning and DQN, the system demonstrates a significant capability to dynamically optimize decoy deployment in response to attacker behavior. Experimental results in a simulated network environment, featuring dynamic attackers, validate the system's effectiveness. Key advantages over traditional methods are highlighted below:

STRONG ADAPTABILITY

The reinforcement learning-based honeypot system is highly adaptive, capable of real-time updates to its defense strategies based on the attacker's behavior patterns and system feedback. This adaptability allows the honeypot system to respond rapidly to new types of attacks without relying on static rules or manual updates. Traditional firewalls and intrusion detection systems typically require regular updates to rules and signature databases, making them slow to respond to advanced threats. In contrast, the flexibility of the honeypot system significantly improves defense efficiency and responsiveness.

DYNAMIC DECEPTION CAPABILITY

The honeypot system also features dynamic deception capabilities. Through reinforcement learning, the system can flexibly adjust the position, number, and types of honeypots, directing attackers to false targets and reducing the potential threat to real systems. This dynamic deception not only misleads attackers but also allows the collection of attack behavior data, providing valuable information for further analysis. Traditional defense systems are mostly passive and lack the ability to proactively deceive attackers, making them relatively reactive when facing diverse attack methods.

OUTSTANDING PERFORMANCE AGAINST APT

The system performs exceptionally well in dealing with APT. APT attacks typically exhibit multi-phase, long-term, and stealthy characteristics. Traditional defense methods, which mainly rely on static rules and signature detection, are vulnerable to being bypassed by APT attacks. Through reinforcement learning, the

honeypot system can progressively identify and adapt to the long-term behaviors of attackers, improving its ability to capture APT attacks. The system can gradually optimize its defense strategies through multiple learning cycles, effectively countering complex and sustained attacks, and overcoming the shortcomings of traditional defense methods in APT detection.

In conclusion, the reinforcement learning-based honeypot system offers significant advantages over traditional defense methods in terms of adaptability, dynamic deception, and defense against APT attacks. Traditional defenses are largely passive and struggle to dynamically adjust to new types of attacks. In contrast, the honeypot system achieves intelligent and proactive defense strategies through reinforcement learning.

For the textile and leather industries, these advancements are of paramount importance. As smart factories, digital supply chains, and connected products become the norm, the ability to proactively defend against sophisticated cyberattacks—such as those aiming to steal proprietary fabric designs, disrupt automated production lines, or compromise supply chain data—is crucial. The proposed adaptive deception defense system provides a robust solution to safeguard these critical digital assets, thereby ensuring operational resilience, protecting intellectual property, and maintaining the competitive advantage of the sector in the digital age.

In the future, as reinforcement learning technology continues to advance, the reinforcement learning-based honeypot system is expected to become a crucial tool for network security defense, particularly in critical industries like textiles and leather, providing more effective solutions for addressing complex network security threats.

Author Contributions

Conceptualization – Q.Z., L.F. and H.M.; methodology – Q.Z., L.F. and L.B.; formal analysis – Q.Z., L.F. and L.X.; investigation – Q.Z. and Z.C.; resources – L.F. and L.B.; writing-original draft preparation – Q.Z., L.F. and H.M.; writing-review and editing – Q.Z. and L.F.; visualization – L.F.; supervision – L.F. All authors have read and agreed to the published version of the manuscript.

Conflicts of Interest

The author(s) declared no potential conflicts of interest with respect to the research, author-ship, and/or publication of this article.

Funding

This work was supported in part by the China Southern Power Grid's major network-level scientific and technological project "Research and Application of Multi-dimensional Active Defense Technology for Digital Grid", grant number 037800KC24040002 (GDKJXM20240428).

Data Sharing Agreement

The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Acknowledgements

Not applicable.

REFERENCES

[1] Abdelrazek L, Fuladi R, Kövér J, Karaçay L, Gülen U. Detecting IP DDoS attacks using 3GPP radio protocols. IEEE Access. 2024; 12:24776-24790. doi: 10.1109/ACCESS.2024.3365425

[2] Yuill JJ. Defensive Computer-Security Deception Operations: Processes, Principles and Techniques. Raleigh, NC, USA: North Carolina State University; 2006. ISBN: 978-0-549-37681-1.

[3] Reti D, Fraunholz D, Elzer K, Schneider D, Schotten HD. Evaluating deception and moving target defense with network attack simulation. In Proceedings of the 9th ACM workshop on moving target defense. 2022 ; 45-53. doi: 10.1145/3560828.3564006

[4] Jajodia S, Ghosh A, Swarup V, Wang C, Wang X, editors. Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats. New York, USA: Springer; 2011. doi: 10.1007/978-1-4614-0977-9

[5] Wu J. Research on cyber mimic defense. Journal of Information Security. 2016; 1(4):1-10. doi: 10.19363/j.cnki.cn10-1380/tn.2016.04.001

[6] Gao C, Wang Y, Xiong X. MTD-enhanced network deception defense system. Journal of Computer Engineering and Applications. 2022; 58(15). doi: 10.3778/j.issn.1002-8331.2105-0169

[7] Jiang W, Fang B-X, Tian Z-H, Zhang H-L. Evaluating network security and optimal active defense based on attack–defense game model. Chinese Journal of Computers. 2009; 32(4):817-827. doi: 10.3724/SP.J.1016.2009.00817

[8] Djamaluddin B, Alnazeer A, Azzedin F. Web deception towards moving target defense. In: Proceedings of the 2018 International Carnahan Conference on Security Technology (ICCST); 22-25 Oct 2018; Montreal, QC, Canada. Piscataway, NJ, USA: IEEE; 2018. p. 1-5. doi: 10.1109/CCST.2018.8585457

[9] Wang J, Zhang Z, Wang M. A trust management method against abnormal behavior of industrial control networks under active defense architecture. IEEE Transactions on Network and Service Management. 2022; 19(3):2549-2572. doi: 10.1109/TNSM.2022.3173398

[10] Clark A, Sun K, Poovendran R. Effectiveness of IP address randomization in decoy-based moving target defense. Proceedings of the 52nd IEEE Conference on Decision and Control (CDC); 10-13 Dec 2013; Florence, Italy. Piscataway, NJ, USA: IEEE; 2013. p. 678-685. doi: 10.1109/CDC.2013.6759960

[11] Sugrim S, Venkatesan S, Youzwak JA, Chiang C-YJ, Chadha R, Albanese M, et al. Measuring the effectiveness of network deception. Proceedings of the 2018 IEEE International Conference on Intelligence and Security Informatics (ISI); 9-11 Nov 2018; Miami, FL, USA. Piscataway, NJ, USA: IEEE; 2018. p. 142-147. doi: 10.1109/ISI.2018.8587326

[12] Ma D, Tang Z, Sun X, Guo L, Wang L, Chen K. Game theory approaches for evaluating the deception-based moving target defense. Proceedings of the 9th ACM Workshop on Moving Target Defense; 7 Nov 2022; Los Angeles, CA, USA. New York, NY, USA: ACM; 2022. p. 67–77. doi: 10.1145/3560828.3563995

[13] Shakarian P, Paulo D, Albanese M, Jajodia S. Keeping intruders at large : A graph-theoretic approach to reducing the probability of successful network intrusions. Proceedings of the 11th International Conference on Security and Cryptography (SECRYPT); 28-30 Aug 2014; Vienna, Austria. Setúbal, Portugal: SCITEPRESS – Science and Technology Publications; 2014. p. 19-30. doi: 10.5220/0005013800190030

[14]Dai W, Li S, Lu L, Ye Y, Meng F, Zhang D. Research on application of mimic defense in industrial control system security. Proceedings of the 2021 IEEE 2nd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA); 17-19 Dec 2021; Chongqing, China. Piscataway, NJ, USA: IEEE; 2021. p. 573-577. doi: 10.1109/ICIBA52610.2021.9688212

[15]Zhao Q, Zhang C, Zhao Z. A decoy chain deployment method based on SDN and NFV against penetration attack. PLoS ONE. 2017; 12(12):e0189095. doi: 10.1371/journal.pone.0189095

[16]Yun T, Luo J, Peng B, Yao D. Dynamic defense methods for endogenously secure industrial control networks. Proceedings of the 2018 Chinese Automation Congress (CAC); 30 Nov - 02 Dec 2018; Xi'an, China. Piscataway, NJ, USA: IEEE; 2018. p. 635-639. doi: 10.1109/CAC.2018.8623141

[17]Gao Y, Zhang G, Xing C. A multiphase dynamic deployment mechanism of virtualized honeypots based on intelligent attack path prediction. Security and Communication Networks. 2021; 2021:1-15. doi: 10.1155/2021/6378218

[18]Miehling E, Rasouli M, Teneketzis D. Optimal defense policies for partially observable spreading processes on Bayesian attack graphs. Proceedings of the Second ACM Workshop on Moving Target Defense; 12 Oct 2015; Denver, CO, USA. New York, NY, USA: ACM; 2015. p. 67-76. doi: 10.1145/2808475.2808482

[19]Wang S, Pei Q, Wang J, Tang G, Zhang Y, Liu X. An intelligent deployment policy for deception resources based on reinforcement learning. IEEE Access. 2020; 8:35792-35804. doi: 10.1109/ACCESS.2020.2974786