Collaborative path penetration in 5G-IoT networks: A multi-agent deep reinforcement learning approach

Hang Shen[1] · Xiang Li[1] · Yan Wang[1] · Tianjing Wang[1] · Guangwei Bai[1]

Received: 13 August 2024 / Accepted: 18 February 2025
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2025

Abstract

The 5th Generation (5G) Mobile Network, coupled with the Internet of Things (IoT), is a heterogeneous environment prone to various security vulnerabilities and frequent attacks. Therefore, analyzing attackers' intrusion intentions and penetration behaviors is crucial for guiding network security defenses. However, existing penetration path constructions mostly rely on a single agent or consider a single type of network, which cannot comprehensively assess the impact of group attack behaviors and system vulnerability combinations on security. To address this issue, a multi-agent reinforcement learning approach is presented for collaborative path penetration. This solution leverages network situational information to guide policy learning, improving the overall path penetration quality. Agents adopt a hierarchical structure of primary and subordinate roles. The primary agent can observe the entire environmental state and formulate top-level collaborative strategies for the group, while subordinate agents learn individual policies based on assigned tasks and team rewards and interact with the environment to learn. Through this mode, agents coordinate with each other to select and continuously improve their strategies in penetration. Experimental results show that compared with single-path penetration under a single agent, the proposed method generates multiple penetration paths with higher attack efficiency, decision stability, and task completion rate.

Keywords Collaborative path penetration · Multi-agent deep reinforcement learning · Primary and subordinate agents

1 Introduction

The 5th-generation wireless systems (5G) not only offer higher bandwidth and lower latency but also support a vast array of devices through the 5G Internet of Things (IoT) [1, 2]. This integration results in an increasingly complex and open network structure, encompassing both conventional user equipment and a multitude of machine-type IoT devices. As these devices become embedded in critical infrastructure-from smart homes to smart cities-the attack surface expands, making 5G IoT networks more susceptible to security threats. The potential vulnerabilities within these networks may lead to significant economic and social repercussions, emphasizing the need for robust security measures [3].

Penetration testing, a vital technique for evaluating network security, plays a crucial role in predicting, preventing, and responding to potential threats within 5G-IoT networks. By simulating real-world attack scenarios, penetration testing helps network administrators understand penetration principles, enabling them to discover system vulnerabilities and potential penetration paths that attackers might exploit. This understanding allows for the design of targeted defense strategies tailored to specific threats [4–6]. Penetration testing can be performed in direct and indirect modes: the direct mode involves an attacker launching an attack from one host to another, while the indirect mode involves compromising an intermediary host (a stepping-stone) to obscure the attacker's

This article is part of the Topical Collection: *Special Issue on 2 - Track on Security and Privacy*
Guest Editor: Rongxing Lu

✉ Tianjing Wang
wangtianjing@njtech.edu.cn

Hang Shen
hshen@njtech.edu.cn

Xiang Li
lixiang131@njtech.edu.cn

Yan Wang
wy8573200@163.com

Guangwei Bai
bai@njtech.edu.cn

[1] College of Computer and Information Engineering (College of Artificial Intelligence), Nanjing Tech University, Nanjing 211816, China

true source and identity, thereby complicating detection [7]. However, the dynamic nature of network environments and the unpredictability of attacker behavior make it challenging to construct effective penetration strategies [8, 9]. Additionally, the continuous evolution of network defense systems necessitates ongoing refinement and adaptation of penetration testing techniques to maintain the security and reliability of 5G-IoT networks.

Deep Reinforcement Learning (DRL), which integrates Reinforcement Learning (RL) with deep learning, has shown exceptional performance in tackling complex sequential decision-making tasks [10]. It has become a popular method for generating penetration paths and developing security strategies [11]. Wang et al. [12] proposed a globally guided RL method that utilizes spatiotemporal information from the environment to guide a mobile robot in making local path adjustments when encountering obstacles without the need to re-invoke the planning algorithm to find an alternative route, thereby improving the model's generalization capability. In the RL method proposed by Cody T et al. [13], the agent learns through environmental interaction to discover multiple attack paths, providing references for defense measures such as inter-subnet router firewalls, authentication log tracking, and host-based antivirus solutions. Zhang et al. [14] proposed a DRL-based multi-domain action selection method for agents to discover more hidden multi-domain penetration paths. In [15], the author developed an automated penetration testing framework utilizing DRL to streamline the process of security vulnerability assessment. This framework aims to emulate attackers to identify the most effective attack paths within a network topology.

Traditionally, research in this area has focused on single-agent decision-making. However, with the rise of heterogeneous networks by 5G-IoT, the landscape has shifted towards sophisticated attacks involving multiple attacks collaborating in complex, coordinated efforts. In these environments, multiple agents interact and learn simultaneously, presenting challenges that fall within the scope of Multi-Agent DRL (MADRL) techniques [16–19].

1.1 Challenging issues

In network penetration testing, the complexity of modern attacks often exceeds the capabilities of a single agent, which struggles to detect and respond effectively to these collaborative efforts due to its limited perceptual range. Therefore, analyzing the behavior patterns of multiple attackers-such as target selection, attack methods, and path planning-becomes essential. By leveraging the MADRL framework, complex tasks can be decomposed into sub-tasks, each managed by individual agents working in cooperation. This approach enhances efficiency and flexibility, providing more comprehensive and targeted security measures. However, designing multi-attacker collaborative penetration strategies for 5G-IoT networks presents significant technical challenges.

1) *Collaborative environmental perception*: 5G-IoT networks are complex, incorporating various custom network protocols, services, diverse operating systems, configurations, and dynamic network topologies. These networks typically consist of multiple heterogeneous agents, each with varying perceptual abilities, computational resources, and communication capabilities [20, 21]. Effectively integrating environmental information from these diverse sources poses a major challenge. The heterogeneity of agents can lead to inconsistent or incomplete perceptions of the environment, which complicates decision-making. Overcoming these challenges requires sophisticated methods for aggregating and processing disparate data streams to ensure the collaborative system can operate with a unified and accurate understanding of the network environment [22].
2) *Environmental information sharing*: In multi-agent systems, timely sharing and updating of environmental information are critical for maintaining coordination [23], especially in adversarial or competitive environments where delays can be exploited by attackers. Agents operating in different locations often have access to only partial information, necessitating a robust exchanging mechanism to achieve a globally consistent situational awareness [24]. However, this process is hindered by potential interaction delays, packet loss, and the limitations imposed by resources [25]. As the number of agents increases, the demands on real-time performance and the reliability of information sharing become bottlenecks, impacting the system's ability to make swift and coordinated decisions. Addressing these issues requires an efficient collaboration framework and protocol.
3) *Coordination and consistency*: In a collaborative penetration scenario, agents involved possess varying capabilities and characteristics, such as differences in mobility, stealth, and aggressiveness [26]. These variations mean that while some agents may prioritize minimizing penetration time, others might focus on maximizing the success rate of the penetration. Coordinating these diverse objectives and ensuring consistency in group decision-making are significant challenges, which require a strategic allocation of penetration tasks that align with each agent's strengths while satisfying the overall attack goals [27]. This necessitates advanced algorithms capable of dynamically adjusting strategies based on real-time feedback and the evolving conditions of the network environment, ensuring that all agents operate cohesively towards a common objective.

1.2 Contributions and organization

To address the aforementioned constraints and challenges, this paper proposes a collaborative penetration method based on the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm. In our study, agents refer to the independent entities performing penetration testing tasks within the 5G-IoT network. Each agent is capable of perceiving its environment, making decisions, and executing specific attack behaviors. The agents coordinate through the Contract Net Protocol (CNP), dynamically bidding for tasks and allocating roles to achieve more efficient task completion. Unlike conventional single-agent path planning, the proposed method can enhance overall penetration path quality through multi-agent collaboration. The main contributions of this study are as follows:

- A multi-agent collaboration method integrating CNP is developed to solve the task allocation problem in penetration testing scenarios. Agents bid for tasks based on their capabilities, while the primary agent assigns tasks to the best agents. During task execution, agents can share real-time local penetration states and attack behaviors, improving collaboration efficiency;
- A prioritized experience replay (PER) mechanism is introduced to overcome the limitations of single-agent

Table 1 Glossary of Technical Terms

Term	Definition
5GC (5G Core Network)	The central component of the 5G network that handles session management, mobility management, policy control, and other functions.
5GC Servers	Servers that support the core functions of the 5G core network, including session management, user mobility, and traffic routing.
Application Core Zone	The network segment that connects the application layer through integrated services such as communication and computing.
Attack Graph (AG)	A directed graph that models vulnerabilities and possible attack paths within a system, with directed edges representing state transitions.
Boundary Firewall	A firewall that protects the edge of a network by controlling traffic between the internal network and external entities.
Contract Net Protocol (CNP)	A task allocation protocol where agents bid for tasks, and the best-suited agent is selected to execute them based on predefined criteria.
CVSS (Common Vulnerability Scoring System)	A standardized system for rating the severity of security vulnerabilities in software and hardware.
DMZ (Demilitarized Zone)	A buffer zone in a network, hosting servers and providing functions like risk isolation, access control, and monitoring of external traffic.
Importance Sampling Weights (ISW)	A method used in MADDPG to adjust the probability of selecting different experiences, ensuring that more important experiences are prioritized.
Internal Firewall	A firewall that protects different segments of the internal network, controlling traffic between various security zones.
Multi-Agent Deep Deterministic Policy Gradient (MADDPG)	A DRL algorithm used for training agents in a multi-agent system.
Next Generation Radio Access Network (NG-RAN)	The 5G network component that connects user devices to the core network, providing wireless communication. It replaces 4G's eNodeBs with gNodeBs and supports high data rates, low latency, and massive connectivity for 5G services.
Penetration Testing	The practice of testing a system for vulnerabilities by simulating attacks to identify security weaknesses.
Prioritized Experience Replay (PER)	A technique in reinforcement learning where experiences that are more valuable or informative are prioritized during training.
Session Management Function (SMF)	A function in the 5G network responsible for tunnel maintenance, IP address allocation, and user plane selection.
User Plane Function (UPF)	A function in the 5G network that handles routing forwarding, policy enforcement, and traffic reporting.

observation, and a global experience pool is constructed to store the interaction experience trajectories of all agents, with each sample assigned different priorities based on experience quality and rarity. High-quality and rare samples are prioritized during training, accelerating policy convergence and improving the stability of path generation;
- Multi-agent penetration path planning experiments and ablation studies under different relationship settings are designed. Extensive simulation results show that the proposed scheme outperforms mainstream DRL baseline methods in terms of environmental adaptability, penetration efficiency, decision stability, and task completion rate.

The remainder of this paper is organized as follows: The 5G-IoT system model and problem formulation are introduced in Section 2. Section 3 presents the MADDPG-based collaborative penetration approach. We explain the experimental preparation in Section 4 and analyze results in Section 5. Section 6 summarizes the work and plans follow-up research. Table 1 summarizes the main technical terms used.

2 System model

2.1 Network scenarios

Consider a multi-attacker collaborative penetration scenario for a 5G-IoT network, shown in Fig. 1, which includes the Demilitarized Zone (DMZ), Next Generation Radio Access Network (NG-RAN) [28], 5G core network (5GC) [29], and application core zone. These areas encompass multiple roles such as attackers, hosts, servers, firewalls, 5GC network elements [30], and 5GC servers. Firewalls are categorized as boundary (Firewall 1) and internal firewalls (Firewalls 2, 3). They divide different interfaces into distinct security zones.

The DMZ constitutes a buffer area for deploying servers and serving functions such as risk isolation, access control, server isolation, and external access. All traffic entering and exiting the internal network must pass through the DMZ, facilitating traffic inspection, filtering, and monitoring to prevent unauthorized access. The DMZ hosts web servers, file servers, etc., to provide services to the outside. These are isolated from other servers in the internal network. Only legitimate external users can access the public services in this zone.

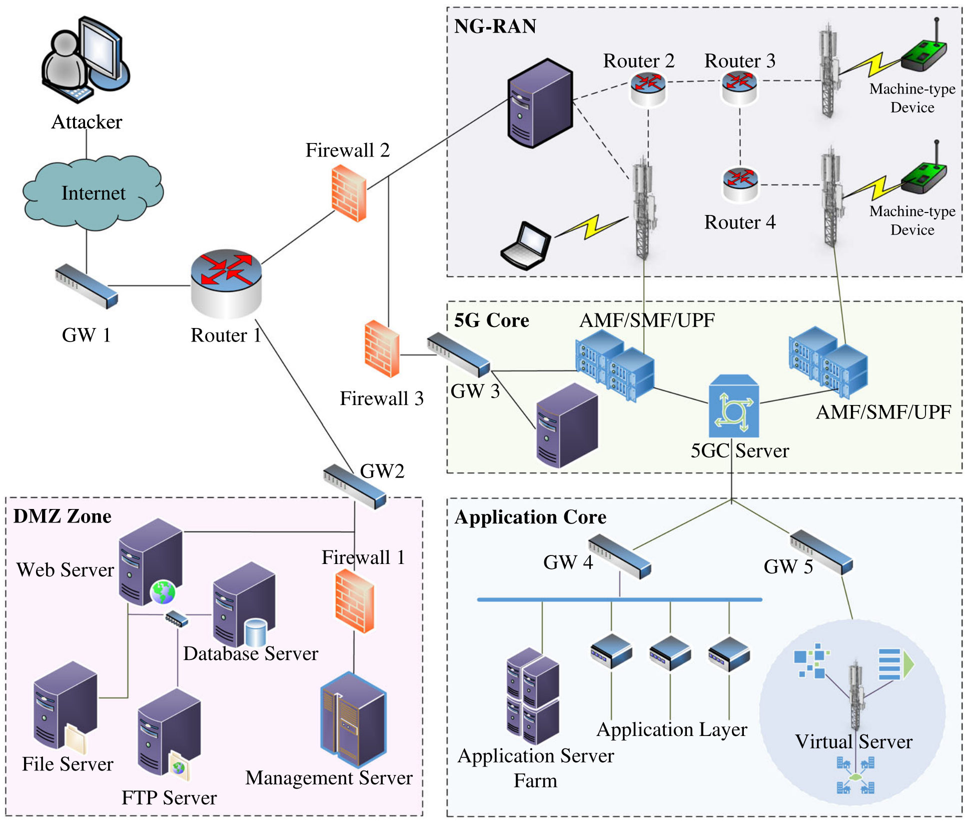


Fig. 1 5G-IoT network topology

Table 2 Vulnerabilities in a 5G-IoT Network

Zone	CVE ID	Score	CVE Description	Published Date
DMZ	CVE-2021-34631	8.8	Plugin is vulnerable to cross-site	08/05/2021
5G Core	CVE-2020-9090	7.8	Insecure search path vulnerability	03/15/2020
NG-RAN	CVE-2018-8652	5.4	A cross-site scripting (XSS) vulnerability exists	11/12/2018
Application Core Zone	CVE-2021-33636	8.4	The attacker can execute arbitrary code	10/29/2023

NG-RAN and 5GC are key components of the 5G-IoT network, responsible for wireless communication with terminal devices and processing wireless protocols. 5GC is the 5G core network, handling session management, mobility management, policy control, and other functions. The Access and Mobility Management Function (AMF) performs registration, connection, and mobility management and provides a transmission channel for session management messages. The Session Management Function (SMF) is responsible for tunnel maintenance, IP address allocation and management, user plane selection, etc. The User Plane Function (UPF) handles routing forwarding, policy implementation, traffic reporting, and other functions.

The application core zone includes both wired and wireless control sites. The former connects to the application layer through Gateway 4, providing integrated services such as communication and computing, while the latter connects to mobile devices via Gateway 5, offering virtual services.

The vulnerabilities used are sourced from the National Vulnerability Database (NVD). As shown in Table 2, we selected vulnerabilities from various network zones within a 5G-IoT environment, including DMZ, 5G Core, NG-RAN, and Application Core Zone. The selected vulnerabilities vary in severity, ranging from medium to high, based on their Common Vulnerability Scoring System (CVSS) scores, with values ranging from 5.4 to 8.8. These vulnerabilities cover a range of attack vectors, including cross-site scripting (XSS) and arbitrary code execution, which are relevant to real-world penetration testing scenarios. We assess the data quality by considering both the severity and difficulty of exploitation. The CVSS score, as shown in Table 2, indicates the vulnerability's severity, with higher scores indicating more critical vulnerabilities that are easier to exploit. For example, vulnerabilities like CVE-2021-34631 (CVSS score 8.8) represent high-risk scenarios.

2.2 Attack graph model

Based on the experimental network topology and the collected host vulnerability information, the AG was generated using the automated tool MulVAL[1].An Attack Graph (AG) is a directed graph, typically represented using an adjacency matrix, that models the relationships between various vulnerabilities in a network. We extracted a portion of the AG from a 5G-IoT scenario for illustrative purposes, specifically focusing on the DMZ and application core areas. In this graph, directed edges (state transitions) are annotated with CVSS scores, which represent the severity of each vulnerability. The CVSS score correlates with the potential attack benefit, where higher scores indicate greater rewards for successful exploitation. This integration helps simulate the effectiveness of agents' strategies in selecting high-value targets. It allows us to quantify the impact of different vulnerabilities on task outcomes and overall agent performance.

The AG in Fig. 2 contains different state nodes, each representing a host state within the network. Node S_1 represents the initial state, and the node labeled Target represents the goal state. The directed edges in the AG signify state transitions from one host state to another, with each edge annotated with the corresponding vulnerability score. These scores reflect the security risk level of transitioning from one host state to another. High scores indicate significant potential threats and a higher likelihood of the vulnerability being exploited by an attacker. Each state transition signifies that the attacker has successfully gained control over that particular node.

In the adjacency matrix, the number of states in the AG determines the matrix's dimensions. In contrast, the scores on the edges represent the values of the corresponding elements in the matrix. For example, when the attacker transitions from state nodes S_5 to S_6, it indicates that the attacker exploited a vulnerability with a score of 8.8 on the host to complete this state transition.

2.3 Problem formulation

The optimal penetration path aims to find a path with maximum attack gain between the initial and target nodes in the AG model. The main indicators for measuring attack gain include:

- **Vulnerability Score:** Vulnerability scores are assessed based on the vulnerability's exploitability, impact, and scope. The score reflects the ease with which a vulnerability can be exploited, with higher scores indicating

[1] https://www.arguslab.org/software/mulval.html

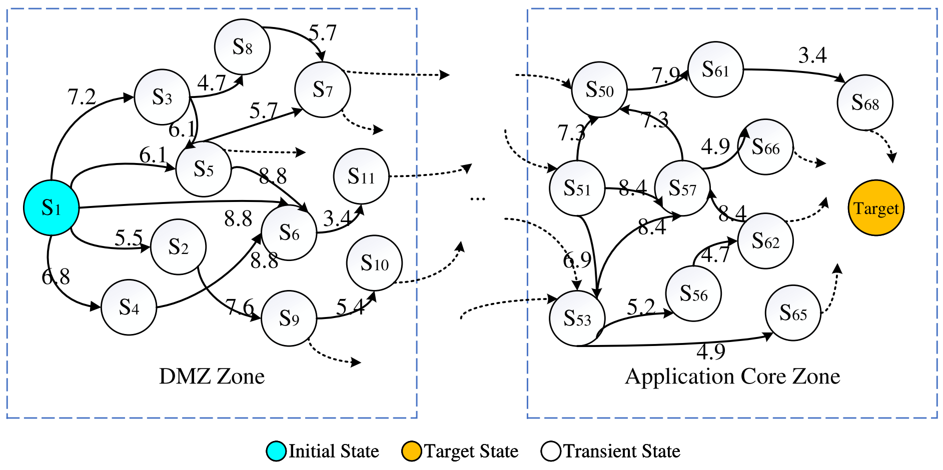


Fig. 2 Attack graph for local scenarios

more effortless exploitation and lower attack difficulty. Attackers often prioritize paths with high vulnerability scores to ensure a high probability of successful attacks.

- **Attack Duration:** Attackers must complete the penetration within a specific time frame; otherwise, defenders may detect attacks and take remedial measures. When determining the optimal penetration path, attackers need to consider the attack time window, aiming to complete the attack relatively quickly to reduce the risk of detection and prevention.

The path between the initial node, N_S, and the target, N_D, in the AG is denoted as $P_{S,D}$, which consists of a sequence of state nodes s_1, ..., s_n, satisfying two conditions: 1) $\forall i$, $1 \leq i \leq N$, $(s_i, s_{i+1}) \in E$; 2) each node only acts once in the path. Let $P^1_{S,D}$ and $P^k_{S,D}(k > 1)$ be the paths obtained for the first and the k-th time, respectively, and $L^1_{S,D}$ and $L^k_{S,D}$ be the path lengths obtained for the first and the k-th time.

Let 0-1 variable $z_{i,j}$ denote whether an attacker transitions from state s_i to state s_j.

$$z_{i,j} = \begin{cases} 1, & \text{attacker moves from } s_i \text{ to } s_j \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The length of an attack path for a penetration task is represented as

$$L_{S,D} = \sum_{(s_i,s_j)\in E} z_{i,j}. \tag{2}$$

The vulnerability score on an edge form s_i to s_j is denoted as $f_1(e_{i,j})$. The attack cost for an attacker to transition from state s_i to state s_j is represented as $f_2(z_{i,j})$, which is used to balance the attack duration and vulnerability score, preventing the system from falling into an infinite loop. $f_2(z_{i,j})$ is with the following properties:

- For each state transition, there is a unique $f_2(z_{i,j})$ corresponding to it;
- When $L^1_{S,D} \geq L^k_{S,D}$, the attack cost is not considered;
- When $L^1_{S,D} < L^k_{S,D}$, $L^k_{S,X}$ represents the partial path length in $P^K_{S,D}$, and $L^k_{S,X} = L^1_{S,D}$, with $N_X = s_m$.

For each subsequent state transition in $P^K_{S,D}$ starting from s_m, the attack cost is defined as

$$f_2(z_{m,j}) \triangleq d^v_{i,j} \tag{3}$$

Let v represent the attack cost exponent, calculated as

$$v = \exp(\frac{\ln H}{E^* - L^1_{S,D}}). \tag{4}$$

where E^* represents the total number of directed edges in the current AG, and H represents the maximum value of the vulnerability score.

The penetration path obtained for the first time is represented as $P^1_{S,D}$. If $L^1_{S,D} \geq L^k_{S,D}$, we have $f_2(z_{i,j}) = 0$. Each time the attacker successfully penetrates a node s_j, and a state transition occurs, i.e., $z_{i,j} = 1$, they gain the vulnerability score $d_{i,j}$ marked on the directed edge $e_{i,j}$ as an attack gain. When $L^1_{S,D} < L^k_{S,D}$ and the attacker continues to launch penetration attacks, both $f_1(e_{i,j})$ and $f_2(z_{i,j})$ need to be considered. Accordingly, the attack gain for the k-th penetration path is defined as

$$g_k \triangleq \begin{cases} \sum\limits_{(i,j)\in E} f_1(e_{i,j}) \cdot z_{i,j}, \text{ if } L^1_{S,D} \geq L^K_{S,D} \\ \sum\limits_{(s_i,s_j)\in E} f_1(e_{i,j}) \cdot z_{i,j} - \sum\limits_{(s_i,s_j)\in E} f_2(z_{i,j}) \cdot z_{i,j}, \\ \text{otherwise.} \end{cases} \tag{5}$$

The optimal strategy for a penetration task is to find a path, $P^k_{S,D}$, in the AG that maximizes the attack gain given $P^1_{S,D}$, formulated as $\mathcal{P}1$.

$$\mathcal{P}1 : \max_{P^k_{S,D}} g_k = x \sum_{(s_i,s_j)\in E} f_1(e_{i,j}) \cdot z_{i,j} + (1-x)\left(\sum_{(s_i,s_j)\in E} d_{i,j} \cdot z_{i,j} - \sum_{(s_i,s_j)\in E} f_2(z_{i,j}) \cdot z_{i,j} \right)$$

$$\text{s.t.} \begin{cases} x = \begin{cases} 1, L^1_{S,D} \geq L^K_{S,D} \\ 0, L^1_{S,D} < L^K_{S,D} \end{cases} & \text{(6a)} \\ \begin{cases} \sum_{j=1}^{N} z_{i,j} = 1, \forall s_i \in P^k_{S,D} \cap i \neq n \\ \sum_{i=1}^{N} z_{i,j} = 1, \forall s_j \in P^k_{S,D} \cap j \neq 1 \end{cases} & \text{(6b)} \end{cases}$$

In Constraint Eq. 6a, 0-1 variable x determines the attack benefit expression according to the value under different conditions. Constraint Eq. 6b ensures that the path obtained the first time satisfies: 1) except for the destination state node, the out-degree of each state node is 1; 2) except for the initial state node, the in-degree of the remaining state nodes is 1.

When facing a large-scale complex network scenario, a single penetration path may not fully expose the system's security vulnerabilities and weak points. Constructing multiple penetration paths helps to comprehensively detect potential risks at various levels, thereby improving defense strategies. Extending from Problem $\mathcal{P}1$, the optimal multi-path collaborative penetration problem is modeled as

$$\mathcal{P}2 : \max_{P^k_{S,D}} : \sum_{k=2}^{N} g_k, \text{ s.t. Eqs. 6a and 6b}$$

to maximize the aggregated attack path gain while satisfying path constraints.

3 Proposed method

Complex decision-making relies on multiple agents learning and processing different tasks simultaneously. However, increasing the number of agents poses challenges to managing their interactions. To address this, coordination and balance are introduced to regulate multi-agent collaboration. Before algorithm design, we analyze the characteristics of different MADRL architectures [31] and then select a suitable base model for 5G-IoT penetration testing. Subsequently, key components of this model are optimized to form an efficient collaborative multi-agent system.

MADRL architectures can be categorized into three types:

- *Centralized training and decision-making (Scheme 1)*: All agents share information during both training and decision-making, which aligns with a cooperative game framework. The agents work together towards a shared goal, with the central controller coordinating their actions. The challenge in this setup is the exponential growth in computational complexity as the number of agents increases. From a game-theoretic point of view, this can be seen as a cooperative game [32, 33] with full information, where the players (agents) are trying to reach a Nash equilibrium [34] that maximizes their collective benefit. However, as the network grows, the equilibrium becomes hard to compute, and the solution becomes computationally expensive.
- *Decentralized training and decision-making (Scheme 2)*: Each agent acts independently, making decisions based solely on local observations. This setup aligns more closely with non-cooperative games, where agents must decide on actions without considering the global state or the actions of other agents. Each agent might try to maximize utility, leading to suboptimal outcomes (a Pareto inefficient Nash equilibrium) because the lack of coordination could prevent the agents from reaching a more optimal collective state.
- *Centralized training and decentralized decision-making (Scheme 3)*: This scheme combines the benefits of centralized training (to learn coordinated strategies) with decentralized execution (to allow independent action during real-time task execution), which can be seen as a cooperative game during training and a non-cooperative game during execution.

Table 3 summarizes the advantages and disadvantages of different categories. Under Scheme 1, as the number of agents increases, the network topology space and the dimensionality of penetration actions grow exponentially, directly leading to a significant increase in computational complexity. Meanwhile, all agents must update their policy functions, which is unsuitable for dynamic network environments. Under Scheme 2, each agent makes decisions autonomously and independently of others. Agents do not share information and make decisions entirely based on local observations. Such uncoordinated behavior may lead to suboptimal penetration attack strategies.

Collaborations should consider: 1) Penetration path planning requires highly coordinated actions among multiple agents, thus necessitating the use of global information during training to learn how to coordinate; 2) Penetration tasks

Table 3 Vulnerability Information

Scheme	Architecture	Training	Decision-making
1	Centralized training - centralized decision-making	Communication required	Communication required
2	Decentralized training - decentralized decision-making	No communication	No communication
3	Centralized training - decentralized decision-making	Communication required	No communication

require agents to maintain stealth and decentralized execution can reduce communication burden, lowering the risk of detection and better simulating attack behavior; 3) During the execution phase, attackers need to make quick attack decisions based on real-time information, and decentralized execution allows for diversity in attacker behavior. From the above analysis, Scheme 3, i.e., centralized training - decentralized decision-making, is more suitable for the scenario under study.

Under Scheme 3 corresponding to Fig. 3, the training of all agents is centrally scheduled by a controller. Each agent's global AG state transition information or penetration actions are shared during training. This sharing enables agents to learn effective strategies, accelerating their training process. However, during the execution phase, each agent is decentralized and does not depend on the central controller. In this phase, each agent independently makes local penetration actions based on observations without knowing the actions or states of other agents. This decentralization helps improve the robustness and scalability of penetration testing.

3.1 Multi-agent collaboration

Following the centralized training and decentralized execution architecture, this subsection designs a multi-Agent-collaborative penetration.

Consider a scenario where M agents jointly initiate penetration. The policy parameters of agent i are represented as θ_i. Agent i corresponds to a policy network $\pi(o_i, \theta_i)$ and a value network $Q(s, a)$. The policy network is deterministic, meaning that for the observed local network topology state o_i, the output penetration action $a_i = \pi(o_i, \theta_i)$ is deterministic. The centralized value network takes as input the current global system state $s = [o_1, o_2, \ldots, o_M]$ and the penetration actions of all agents $a = [a_1, \cdots, a_M]$, outputting the quality of executing penetration actions in the current state.

The collaborative penetration based on MADDPG uses the following information during training: the current record in the experience replay pool, all M policy networks, and the i-th value network. As shown in Fig. 4, all agents share a centralized value network (critic-network). This network can

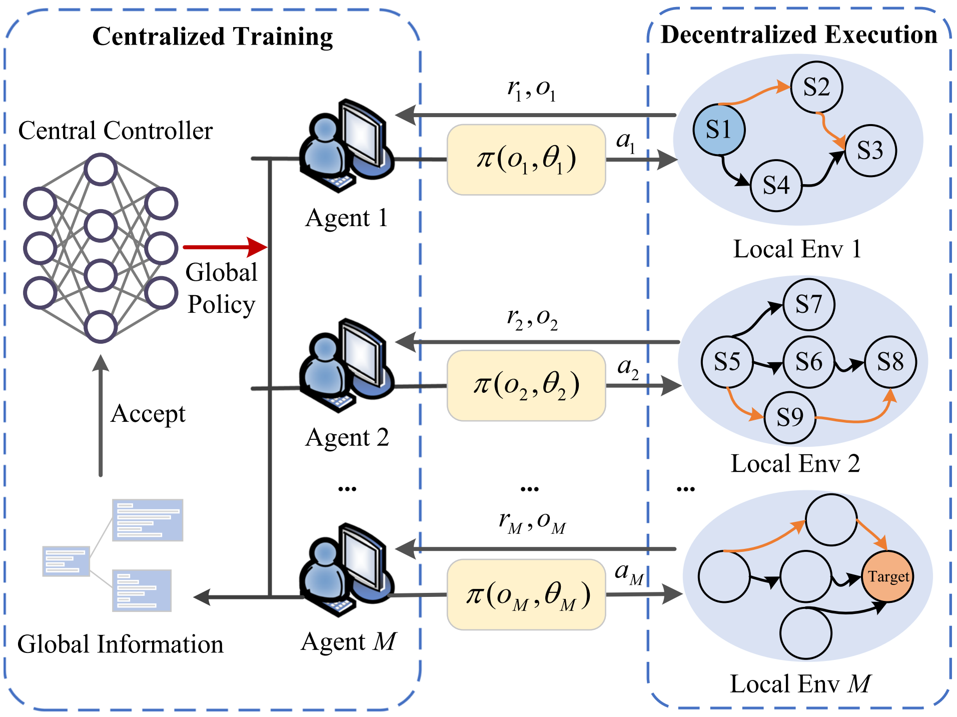


Fig. 3 Joint architecture of centralized training and decentralized execution

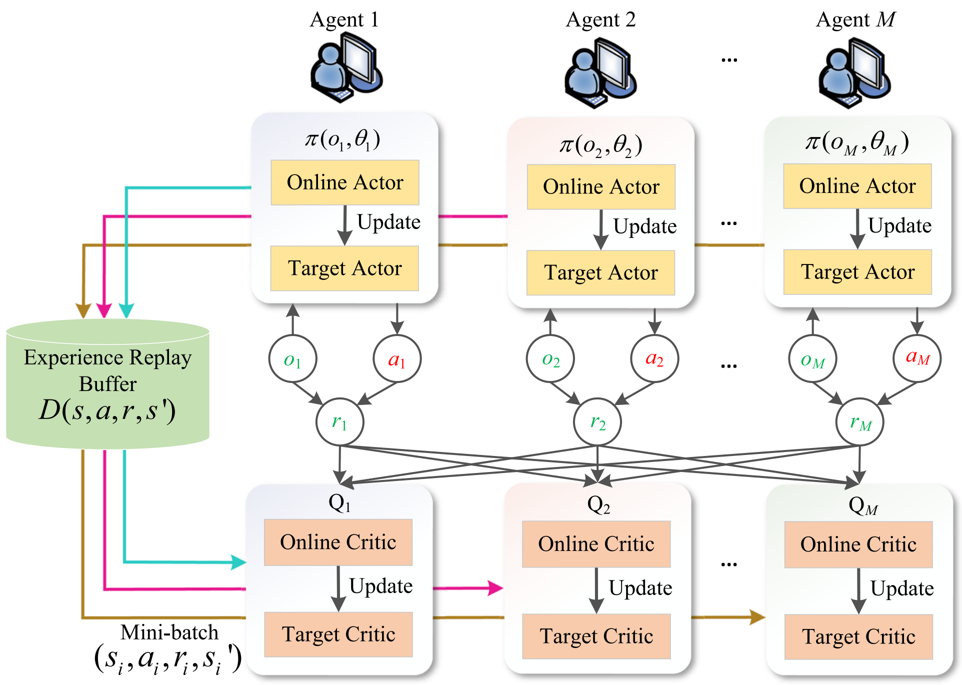


Fig. 4 MADDPG training and execution process

guide each agent's policy network (actor-network) during model training, while during execution, each agent's actor-network operates completely independently [35, 36].

The central controller trains and deploys all attackers' policy and value networks for multi-attacker-collaborative penetration. During the training phase, each attacker's policy network is deployed on the central controller, executing actions according to the controller's instructions. Meanwhile, the value network is used to evaluate the quality of each action. After training is completed, the value network is no longer needed for judgment, and only the policy network is required for decision-making, which is then deployed to the corresponding agent. During penetration, attacker i makes attack behavior decisions independently using the policy network deployed locally based on their local observations without instructions from the central controller.

For deterministic policies, the centralized action-value function can be updated according to the following loss function

$$\begin{aligned} L(\theta_i) &= E_{s,a,r,s'}[(Q_i^{\pi}(s,a_1,\cdots,a_M)-y)^2], \\ y &= r_i+\gamma Q_i^{\pi}(s',a_1',\cdots,a_M')|_{a_j'=\pi_j'(o_j)} \end{aligned} \quad (7)$$

where θ_i represents the parameters of the policy network for agent i and $\pi = (\pi'_{\theta_1},\cdots,\pi'_{\theta_M})$ is the set consisting of M continuous target policies used in updating the value function. y denotes the target value, defined as

$$y = r_i+\gamma Q_i^{\pi}(s',a_1',\cdots,a_M')|_{a_j'=\pi_j'(o_j)} \quad (8)$$

where r_i is the reward received by agent i and γ is the discount factor. The parameters θ_i of the policy network are then updated using gradient ascent based on the calculated loss, which ensures that the policy is adjusted to maximize the expected reward. The value network is updated similarly using the critic network. The updates are performed after each training iteration by minimizing the mean squared error between the predicted and target values calculated using the current policy. This process is repeated for multiple episodes, allowing the policy and value networks to learn from the accumulated experiences in the experience replay buffer. A mini-batch approach for sampling experiences helps stabilize the updates and ensures efficient learning. The procedure of MADDPG training for collaborative penetration is shown in Algorithm 1.

3.2 CNP-based task allocation

In the proposed framework, the central controller is responsible for grasping global information and formulating overall strategies. Such a controller can be viewed as a primary agent whose value network must accept all sub-agents' state observations and actions and make a global value assessment of the entire environment's state.

Algorithm 1 Training Algorithm for Collaborative Penetration.

```
Input: Current network environment parameters and the
       corresponding AG
Output: Optimal policy $\pi$ for the current network environment
1  Initialize actor and critic networks for each agent;
2  for episode $\leftarrow$ 1 to MAX_EPISODES do
3      Initialize a random process $\sigma$ for action exploration;
4      for training round $k \leftarrow 1$ to $K$ do
5          for agent $i$ do
6              $a_i \leftarrow \pi(o_i, \theta_i) + \sigma_t$ according to the policy
               network;
7          $a \leftarrow [a_1, \cdots, a_M]$, observe reward $r$ and new state $s'$;
8          Store $(s, a, r, s')$ in experience replay pool $D$;
9          Randomly sample from $D$;
10         for agent $i$ do
11             Centralized training of critic network;
12             Train actor network;
13             Update target actor and critic networks;
14 return $\pi$
```

The CNP, in combination with MADDPG, solves the problems of penetration task allocation and attack action execution in distributed systems. In this context, a task refers to a penetration testing action that needs to be performed by an agent in a network. The goal of using CNP is its well-established ability to efficiently allocate tasks in a decentralized manner while maintaining a high level of coordination among agents. Our multi-agent system employs CNP to assign tasks based on agent capabilities and environmental conditions. The bidding mechanism of CNP allows agents to express their willingness and ability to perform tasks, ensuring that tasks are allocated to the most suitable agent, fitting well within the hierarchical structure of our proposed method.

CNP's agent roles include primary attackers (as central controllers) and collaborative attackers, simulating a distributed task bidding and allocation process, allowing dynamic selection of the most suitable executor to complete tasks. Figure 5 illustrates the task allocation process of this algorithm, which consists of the following stages:

① **Task publishing and bidding:** The primary attacker broadcasts the penetration task details to all collaborative attackers. Each agent submits a bid based on its capabilities (e.g., resources, expertise) and the task requirements, outlining their approach, estimated time, and potential risks. For example, an agent with firewall expertise may highlight its success rate in bypassing firewall defenses.

② **Winning bid and contract signing:** The primary attacker evaluates bids based on criteria such as vulnerability scores, system risks, time costs, and resources. The most suitable agent is selected, and a contract is signed specifying execution details and rewards. For example, the contract may specify success criteria and the reward for exploiting a router's vulnerability.

③ **Task execution:** The selected collaborative attacker begins executing the task by planning the penetration strategy, adjusting tactics to maximize efficiency and ensure the task's success. Other agents continue to bid on subsequent tasks. For example, while one agent executes the attack, others may prepare for follow-up actions like data exfiltration.

④ **Task completion and result feedback:** Once the task is completed, the collaborative attacker submits results to the primary attacker for verification. If successful, the

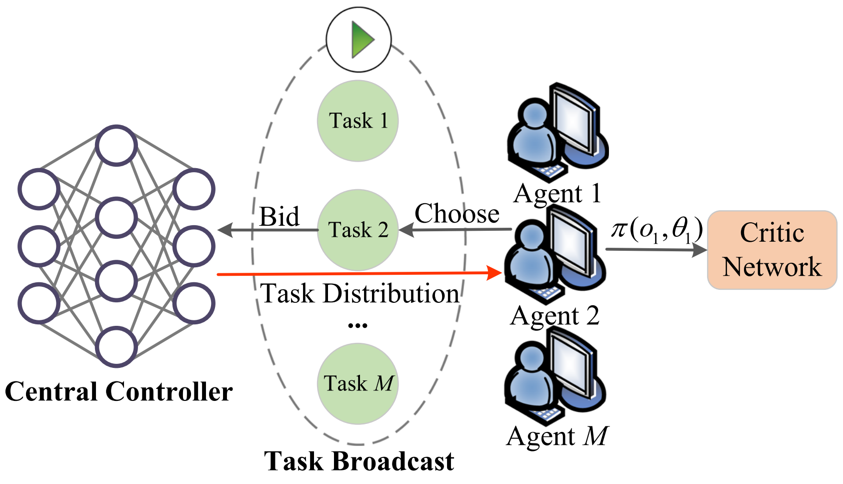


Fig. 5 CNP-based task allocation process

reward is issued; if not, penalties are applied. For example, if a vulnerability is exploited and access gained, the task is completed, and the reward is provided.

Relying on MADDPG, the primary attacker can output penetration task descriptions through the policy network as input to CNP. All agents can participate in bidding, and the winning agent can execute the corresponding policy network for penetration behavior selection based on the bidding results. This explicit division of labor improves overall collaboration efficiency and enhances the system's flexibility and scalability. The procedure of CNP task allocation is summarized as Algorithm 2.

The CNP communication cost includes task announcement, bidding, and task allocation. In the task announcement phase, the central controller broadcasts tasks to all agents, resulting in a communication cost of $O(M)$, where M is the number of agents. During the bidding phase, each agent evaluates the task and sends back a bid, incurring another $O(M)$ communication cost. The task allocation phase involves the central controller assigning the task to the winning agent, which has a communication cost of $O(1)$. As the task complexity increases, the number of bidding rounds (k) may grow, leading to a total communication cost of approximately $O(k \times M)$, where k depends on the task's complexity and the dynamic environment. Accordingly, for a given task, the communication cost can be defined as

$$C \triangleq \mu_1 N + \mu_2 k N + \mathcal{N}(0, \sigma^2) \tag{9}$$

where μ_1 is a constant representing the communication overhead per agent (task announcement and basic communication), μ_2 is a constant representing the communication overhead per bidding round (inter-agent communication for bidding). $\mathcal{N}(0, \sigma^2)$ represents the random noise and simulates the communication fluctuations that may occur in reality, such as network delays or other uncertain factors. The value of Eq. 9 scales linearly with the number of agents and bidding rounds.

The CNP provides a standardized primary-secondary interaction framework for MADDPG. Combining CNP and MADDPG can optimize agents' path planning and task collaboration in penetration testing. Specifically, this protocol subdivides penetration tasks and then assigns subtasks to different individuals according to each agent's self-matching degree. This task allocation method can fully leverage the expertise of each agent, improving penetration efficiency. The communication mechanism among agents in CNP can help coordinate agent behaviors, avoiding conflicts and resource waste. For example, when an agent successfully penetrates a node, it notifies other agents to avoid that node and concentrate resources on new targets.

Algorithm 2 CNP-based Task Allocation Algorithm.

```
   Input: Network environment parameters and the corresponding
          AG
   Output: Value assessment $Q_i^\pi$ of the current subtask
1  Initialize actor and critic networks for each agent;
2  for episode ← 1 to MAX_EPISODES do
3      Initialize a task list task_list ;
4      while $k = 1$ to $K$ and task_list ≠ NULL do
5          Broadcast task and its description, set bidding
           deadline;
6          Wait to receive bids from collaborative attackers;
7          bid_list ← bid;
8          At the bidding deadline, Select collaborative attacker $i$
           with the highest score;
9          Assign the task to collaborative attacker $i$ and notify
           all attackers of the result;
10         Remove assigned task from task_list;
11         Feedback results to the primary attacker, obtain value
           assessment;
12     Continue bidding for the next task;
13 return Value assessment set $Q(s, a)$;
```

3.3 PER mechanism

In the MADDPG algorithm, the experience replay pool can break the temporal correlation of samples, improving the independence and efficiency of sample utilization. This section implements the Prioritized Experience Replay (PER) mechanism to improve the efficiency of penetration path strategy generation by extracting samples with more learning value. By assigning different importance weights to experience samples to guide sample extraction, a prioritized replay can help agents learn efficiently.

Our approach identifies high-quality samples using the Temporal Difference (TD) error, which measures the discrepancy between the predicted rewards and the actual outcomes. In the prioritized experience replay algorithm, TD typically assigns weights to experience samples to learn more frequently from important (i.e., with significant errors) experiences. The following will introduce the calculation for TD and priority.

Suppose an agent chooses a penetration action a_t in state s_t, receives a reward r_{t+1}, and the state transitions to s_{t+1}. The TD error can be represented as

$$\delta_t = r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t). \tag{10}$$

In Eq. 10, γ is a discount factor, determining the importance of future rewards relative to immediate rewards; $Q(s_t, a_t)$ is the current value function estimate for the attack action; $\max_a Q(s_{t+1}, a)$ is the estimate of the maximum return for the next state transition.

Priority calculation is the absolute value of TD plus a small positive constant to avoid zero priority. Samples with larger weights have a higher probability of being selected for learning. The priority weight p_t can be expressed as

$$p_t = |\delta_t| + \varepsilon. \tag{11}$$

ε is a small positive integer, so each experience sample has a certain probability of being selected.

In PER, priority weights are used to change the probability of sample extraction. Typically, priority weights are proportional to the magnitude of TD, meaning samples with larger errors are considered to have higher learning value and are given higher priority. This leads to frequent extraction of high-priority samples during training. To address this, Importance Sampling Weights (ISW) are introduced, which, together with priority weights, adjust prioritized experience replay to compensate for model bias caused by priority sampling.

ISW is inversely proportional to priority. That is, if a sample has a very high priority, its probability of being selected is large, but to avoid this sample having too much influence on the learning process, its ISW should be relatively small. Conversely, if a sample has a lower priority, its probability of being selected is small, and its ISW should be more significant to ensure that when this sample is chosen occasionally, it can make a sufficient contribution to the learning process.

Before calculating ISW, the sampling probability of each experience is represented as

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \tag{12}$$

where α is a hyperparameter in prioritized replay used to adjust the impact of priority, determining how the priority of experience affects the sampling probability. When $\alpha = 0$, sampling becomes random mode. The ISW of a sampled experience sample i is calculated as

$$\omega_i = \left(\frac{1}{Y \cdot P(i)}\right)^\beta \tag{13}$$

where Y represents the total number of experiences in the replay pool, and β is another hyperparameter controlling the weight's influence, usually gradually increasing to 1 over time.

To limit the influence of weights on learning, ISW is normalized within a batch, i.e., each weight is divided by the maximum of all weights in the batch. This normalization ensures that even in the case of non-uniform sampling. ISW adjusts the loss function, thereby updating network parameters through backpropagation. This method balances the contribution of each sample to the gradient based on the probability of the sample being extracted, reducing the estimation bias introduced by priority sampling and helping to stabilize learning.

The PER implementation details are summarized as Algorithm 3, in which rare samples are experiences that occur infrequently within the training dataset. These samples are valuable because they represent unique states or transitions that the agent may not encounter often, helping to improve the agent's generalization capabilities across diverse scenarios. The rarity of samples can influence their priority in the experience replay buffer. Rare samples may be assigned higher priority due to their unique contribution to the agent's learning. As a result, when these rare samples are selected more frequently during training, their ISW is calculated to balance their influence on the learning process, which ensures the model learns effectively from common and rare samples, maintaining a robust learning trajectory. Note that when the experience replay buffer is at capacity, outdated experiences must be discarded promptly. Another is that the initial values and subsequent updates of hyperparameters affect the algorithm's sampling probability distribution density and control weights.

Algorithm 3 Prioritized Experience Replay Algorithm.

```
   Input: Current policy and experience replay memory
   Output: Optimal policy $\pi$ for the current network environment
1  for episode $\leftarrow$ 1 to MAX_EPISODES do
2      Each agent selects an action $a_t$ based on the current policy;
3      Execute joint action $a$, observe reward $r$ and new state $s'$;
4      Calculate TD error $\delta$ and store $(s, a, r, s', \delta)$ in PER;
5      if PER is sufficiently large then
6          for training iteration $k = 1$ to $K$ do
7              Calculate priority $p_i$ for experience sample $i$;
8              Sample a batch $B$ from PER based on priorities
               $p_i$;
9              Calculate loss;
10             Compute $\omega_i$ for sample $i$;
11             Adjust loss using ISW and perform gradient
               descent to update agent parameters;
12             Update Critic and Actor networks;
13             Update TD errors and priorities of samples in
               PER;
14 return $\pi$
```

In a multi-agent penetration testing system, each agent has specific tasks and objectives. Its individual utility refers to the total reward an agent receives from completing these tasks. In a cooperative system, agents' individual utilities must align with the global objective, such as successfully penetrating a target network. While collaboration is necessary for success, agents may still have personal incentives, which could conflict with the collective goal. Balancing these personal goals with team coordination is key to efficient system performance. To address this, CNP is used to ensure agents bid for tasks based on both their capabilities and individual incentives, promoting collaboration while accounting for personal rewards. Additionally, the integration of PER encourages agents to explore and learn strategies that balance individual utility and global outcomes, optimizing both agent performance and overall task success.

4 Experimental preparation

In our experiments, we use a series of simulation settings designed to evaluate the performance of the proposed method and compare it with several baseline methods. The primary goal is to assess the efficiency of multi-agent collaboration in path penetration tasks, focusing on task completion, reward accumulation, and system stability under different relationship settings:

- **Cooperative**: Agents share information and coordinate actions to maximize the system's performance. Each agent's success is aligned with the group's success, and collaboration is encouraged to achieve the best outcome for all involved.
- **Egoistic**: Agents focus on maximizing their rewards without regard for the impact of their actions on the other agents. There is no cooperation, and each agent acts independently, often at the expense of others.

4.1 Simulation setup

The simulation involves a multi-agent system where each agent completes penetration paths in a 5G-IoT network. Agents collaborate and share real-time states during the task execution phase, allowing for dynamic bidding for tasks and role allocation using the CNP. The simulation environment is constructed to test the agents' ability to adapt to network changes and varying levels of cooperation. The communication channels are assumed to support bid submissions, task allocation, and real-time information sharing. These channels are capable of transmitting task details, updates on execution status, and feedback on task completion in real-time. The network is designed to handle these interactions efficiently, ensuring that agents can share local penetration states and attack behaviors without significant delay

The number of agents, N, ranges from 2 to 10, with the number of bidding rounds, k, varying between 1 and 5. Additionally, the base communication overhead, μ_1, per agent is set to 10, with the cost per bidding round, μ_2, set to 5. Random noise is introduced to model environmental variability, simulating real-world network conditions. The detailed parameter settings are given in Table 4.

Table 4 Simulation Settings

Parameter	Value
Number of agents (N)	[2, 10]
Number of bidding rounds (k)	[1, 5]
Base communication overhead (μ_1)	0.6
Cost per bidding round (μ_2)	5
Discount factor (γ)	0.9
Priority weight (p_t)	0.01
Priority adjustment (α)	0.1

4.2 Baseline methods

For comprehensive and intuitive comparison, representative baseline methods are selected/developed, including:

- **Baseline-1:** MADDPG under egoistic relationship setting, where each agent independently seeks to maximize its interests without considering the impact of other agents' actions on the overall system performance. This setting models a scenario where agents act selfishly, focusing solely on their rewards.
- **Baseline-2:** MADDPG under competitive relationship setting, where each agent focuses on maximizing rewards, and agents are in a completely adversarial relationship. During execution, agents hinder each other from achieving their goals;
- **Baseline-3:** Advantage Actor-Critic (A2C) method [37] under a cooperative relationship setting. A2C is a well-known Actor-Critic method that does not incorporate an experience replay mechanism, thus serving as a baseline for comparison with DRL methods that use experience replay.
- **Baseline-4:** MADDPG without CNP and PER, which helps assess their impact on the overall performance.
- **Baseline-5:** MADDPG incorporating CNP to examine the effectiveness of introducing the CNP for task coordination and agent collaboration during training.
- **Baseline-6:** MADDPG with PER [38], to evaluate the impact of Prioritized Experience Replay, which prioritizes experiences with higher learning value, on the learning efficiency and final performance of the agents.

z

4.3 Hyperparameters settings

The hyperparameters γ, ε, and α as defined in Eqs. 8, 11, and 12 are set to 0.9, 0.01, and 0.1, respectively. These values are chosen to balance the trade-off between exploration and exploitation, as well as to control the learning rate, ensuring stable and efficient training across different experimental settings. Specifically, γ represents the discount factor, which determines how much future rewards are valued relative to immediate rewards. A value of $\gamma = 0.9$ means that the agent places moderate importance on long-term rewards while still considering short-term benefits. In Eq. 11, the priority weight p_t is calculated as the absolute value of the TD error δ_t plus a small constant ε (set to 0.01) to prevent zero priority. This ensures that every experience sample has a non-zero probability of being selected for training, which helps avoid neglecting valuable experiences with small or zero TD errors. Lastly, α in Eq. 12 adjusts the impact of priority on the sampling probability in prioritized experience replay. By setting $\alpha = 0.1$, the priority's influence is moderated, preventing overly biased sampling towards highly prioritized experiences and encouraging a more balanced exploration of the experience pool.

4.4 Quantitative metrics

To evaluate the performance of the proposed method, we use the following key metrics:

- **Average Reward:** Measures the cumulative reward agents achieve over time, indicating the system's effectiveness in completing penetration tasks.
- **Task Completion Rate:** The percentage of completed tasks, reflecting agents' efficiency in achieving objectives.
- **Failure Rate:** Tracks the rate of task failures, with lower failure rates indicating better collaboration and coordination among agents.
- **Attack Benefit:** Quantifies the value of penetration paths, considering both quality and length, with higher values indicating more successful exploitation of network resources.
- **Communication Cost:** Assesses task allocation and coordination overhead, with lower costs suggesting efficient use of CNP.

5 Result analysis

This section examines the performance of the proposed approach under different relationships (including cooperative, competitive, and egoistic). Additionally, ablation experiments are designed to analyze the impact of each strategy on overall performance.

5.1 Impact of cooperation modes

In penetration testing scenarios, agents share the goal of successfully penetrating target hosts and gaining privileges. Therefore, the previously proposed MPDRL penetration path planning method assumes a cooperative relationship. This subsection sets up comparative experiments for MADRL applied to penetration path planning under competitive and egoistic relationship settings and compares it with the A2C method under another cooperative relationship setting. In a competitive relationship setting, agents compete for resources in the environment. For example, once agent i successfully penetrates a jump host, other agents will no longer attack that host. In the egoistic relationship setting, each agent tries to maximize its cumulative reward, while the agent's actions change the environment state, benefiting or harming other agents.

We first examine the impact of training iterations on average reward. As shown in Fig. 6(a), the average reward value under the cooperative relationship setting shows a slow growth trend. With the increase in training iterations, the policy learning process tends to stabilize. The rewards fluctuate within a small range of positive values. After incorporating CNP and PER, the main agent and sub-agents can perform efficient task allocation and delivery operations. At

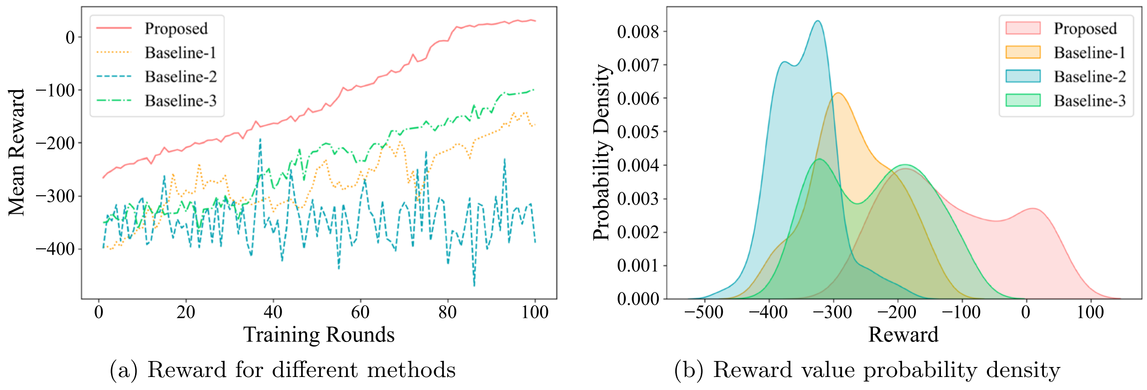


Fig. 6 Rewards and probability density distribution

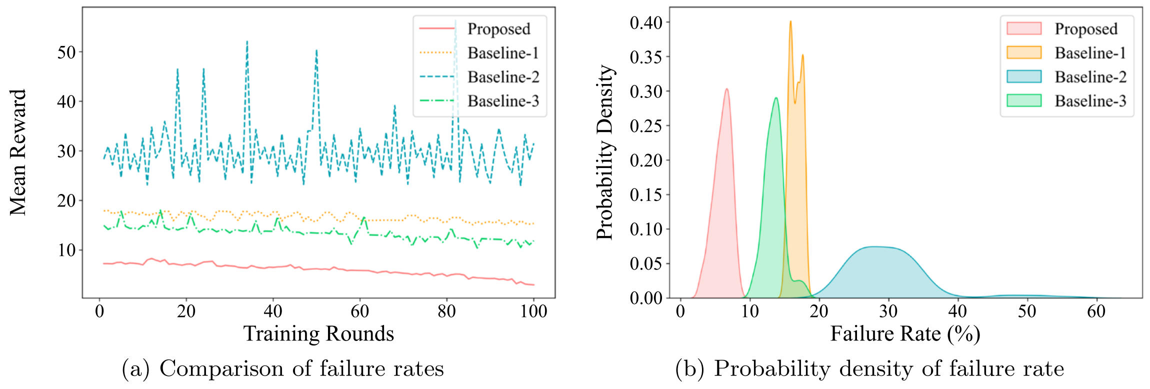


Fig. 7 Failure rate and probability density distribution

the same time, the model can prioritize learning experiences that are critical to accelerating the learning process. Therefore, the proposed method achieves a higher average reward than A2C. The multi-agent system under the egoistic relationship shows a specific growth trend in rewards, but this growth is at a low level and unstable. The rewards under the competitive relationship fluctuate considerably and converge unstably. Figure 6(b) shows the distribution of the four methods' reward value probability density. The proposed method can learn stable strategies. Its distribution is shifted to the right, meaning the rewards are high.

Figure 7 shows the task failure rates and their probability density distributions under three relationship settings. The results under cooperative and egoistic relationship settings show a slow downward trend as training iterations increase. Under the cooperative relationship setting, the task failure rates of the proposed method and Baseline-3 are generally low, confirming the collaboration mechanism's effectiveness in handling complex penetration tasks. Agents under cooperative relationships can share information and learn coordinated strategies, thereby learning and formulating adaptive penetration paths. Under the competitive relationship, significant fluctuations in failure rates are observed, caused by resource contention behavior among agents. In this relationship, agents tend to compete to maximize individual benefits, leading to unreasonable allocation of system resources. Consequently, in some cases, the agents' behaviors conflict, preventing the penetration strategy from converging to the overall optimum.

From the above analysis, cooperation among agents should be encouraged rather than pure self-interest or competition when designing multi-agent systems to address network penetration. Promoting agent collaboration can improve the system's adaptability and robustness and reduce failure rates. During task execution, agents can share real-time local penetration states and attack behavior information. This sharing occurs regularly whenever significant changes happen in the local environment or upon the completion of specific actions, allowing agents to maintain an updated understanding of the overall network situation and adapt their strategies accordingly.

Figure 8 reflects the attack benefit values corresponding to penetration paths generated by multi-agent systems under three relationship settings at different penetration path lengths. The penetration paths generated by the proposed method and baseline method three under the cooperative relationship setting have higher attack benefit values. In contrast, the penetration paths generated by baseline-1 and -2 under the other two relationship settings have lower reference values. This excellent performance is due to two aspects. Cooperative relationships facilitate information sharing and strategy coordination, enabling agents to discover valuable attack targets and paths. On the other hand, centralized training using global information helps comprehensively evaluate and optimize path quality. Distributed execution is suitable for handling large-scale and complex network topologies. In contrast, multi-agent systems under competitive and egoistic relationship settings find it difficult to discover hidden high-value targets due to a lack of cooperation and information sharing. Thus, the attack benefits of generated penetration paths are generally low, and their reference value is relatively limited.

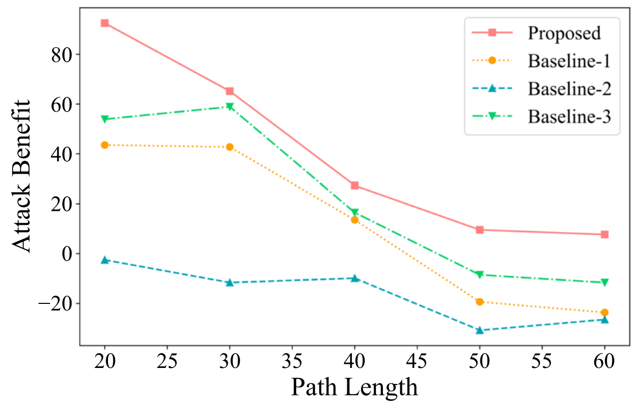


Fig. 8 Attack benefit

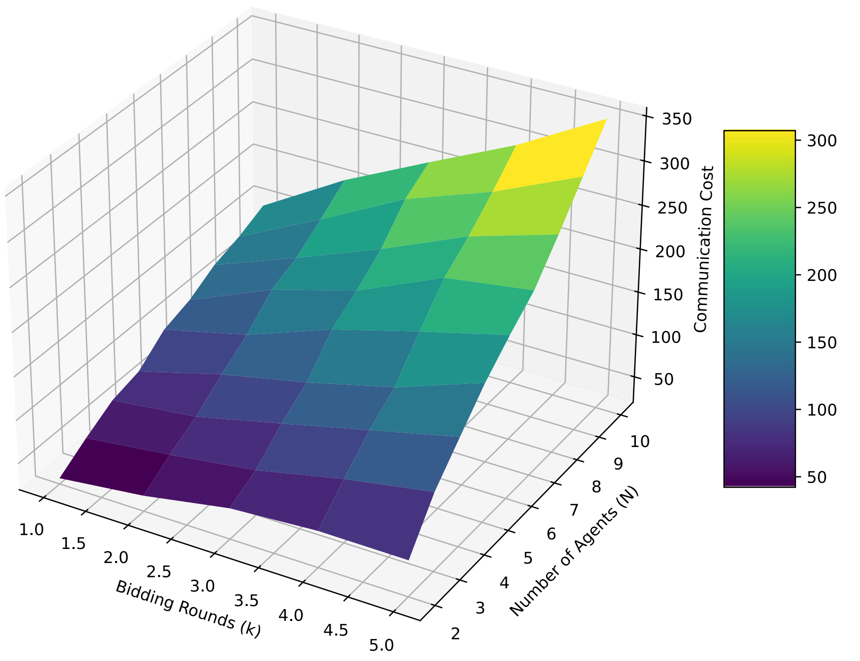


Fig. 9 CNP communication costs in multi-agent systems

To evaluate CNP communication costs in multi-agent systems, we designed a theoretical model and introduced random noise to simulate the variability observed in real-world environments. The number of agents N ranges from 2 to 10, while the number of bidding rounds k ranges from 1 to 5. Each agent's base communication cost is set to 10, and the communication cost per bidding round is set to 5. $\mathcal{N}(0, \sigma^2)$ represents random noise drawn from a normal distribution with a mean of 0 and a standard deviation of 5. From Fig. 9, we observe that the communication cost increases linearly with the number of agents and bidding rounds. As N increases, the communication cost also increases. This is consistent with the nature of the CNP protocol, where each agent must communicate during task allocation. As more agents are added, the communication overhead for task announcement and bidding rises, leading to a linear increase in communication cost. The increase in k also increases communication costs. More bidding rounds mean more agent interactions, resulting in additional communication overhead. Specifically, when the number of bidding rounds increases from 1 to 5, the communication cost increases significantly.

5.2 Ablation experiments

First, we compare the impact of the number of vulnerabilities on task completion rates. Figure 10 shows the task completion rates under different numbers of vulnerabilities. As the number of vulnerabilities increases, the task completion rates of all methods show a certain degree of decline, but the proposed method maintains high task completion rates under different numbers of vulnerabilities. Firstly, the proposed method uses global information to guide penetration behavior, ensuring high execution efficiency; secondly, the unique penetration collaboration promotes mutual understanding and efficient decision-making among agents; finally, the advantage function estimation under the Actor-Critic framework can balance each agent's immediate and long-term returns. Even in complex scenarios with many vulnerabilities, the proposed method can avoid risks and complete expected tasks through coordination and cooperation among agents.

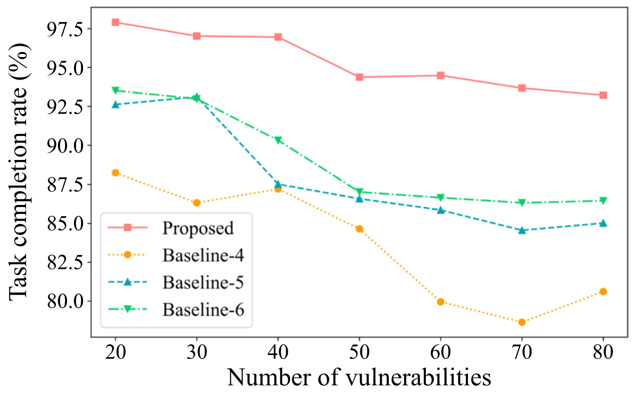


Fig. 10 Task completion rate

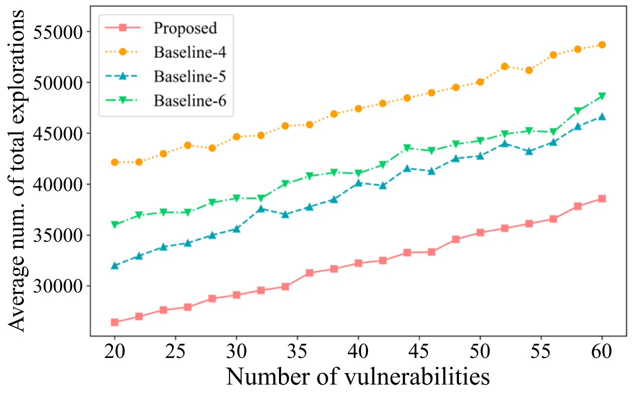


Fig. 11 Path exploration efficiency (Exploration cost for a given number of vulnerabilities)

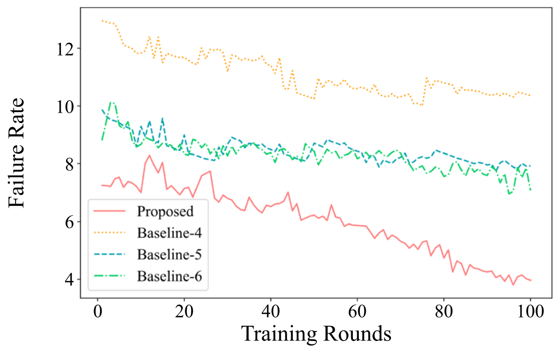
Next, we observe the impact of the number of vulnerabilities on path exploration. Lower exploration steps indicate better overall performance (including learning efficiency, sample utilization, environmental adaptability, determinism, and computational efficiency). As shown in Fig. 11, the exploration steps of each method are positively correlated with the number of vulnerabilities, but the exploration steps of the proposed method are significantly lower than those of the baseline methods, indicating that it can converge to the optimal strategy more quickly. Under the CNP task allocation algorithm, the main control agent can issue requirements based on the current state and accept other agents' commitments, efficiently allocating tasks and using resources. The priority sorting of samples in the experience replay pool ensures that each agent can obtain the most valuable learning samples, and the model's evaluation of the current state and actions is more certain and stable.

Figure 12 shows each method's failure rates and probability density distributions under ablation experiments. The joint optimization helps agents learn mutually beneficial and promoting behavioral strategies in a cooperative environment. Collaboration can promote information sharing and resource utilization among agents through centralized training, joint optimization, and experience replay, enabling agents to capture complex attack and defense dynamics, thereby improving the specificity and robustness of path planning. In 5G-IoT network penetration testing, CNP optimizes task allocation by enabling agents to bid for tasks based on their capabilities, improving coordination and task completion. PER accelerates learning by prioritizing valuable experiences and enhancing policy convergence. Together, they enhance multi-agent collaboration and the effectiveness of penetration strategies.

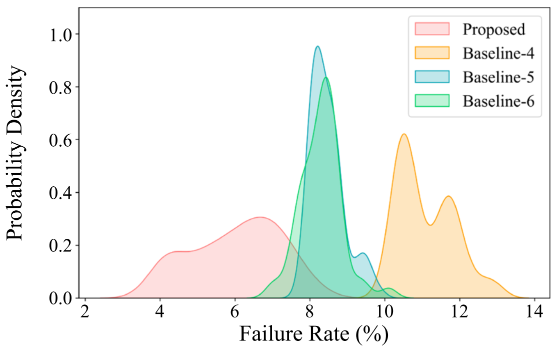
6 Conclusion

A collaborative penetration approach based on MADRL has been proposed in this study. This solution leverages the cooperation of multiple attack agents to enhance the overall efficiency of attack path generation and defense strategy precision. The algorithm is built on centralized training and distributed execution, facilitating multi-agent collaboration to develop and implement precise path planning and penetration strategies. By integrating CNP, the central controller improves task allocation during training, ensuring orderly collaboration among agents. The PER mechanism enhances the directional learning of penetration strategies. The proposed solution offers significant improvements over existing techniques for penetration testing in 5G-IoT networks. While traditional methods focus on single-agent path planning, which limits the scalability and flexibility in dynamic environments, the proposed approach leverages multi-agent collaboration to enhance penetration path quality and efficiency.

Our ongoing work explores the interaction among multiple agents, allowing different agents to assume distinct roles and tasks in penetration.



(a) Failure rates with training rounds.



(b) Probability density of failure rate

Fig. 12 Failure rate and probability density distribution (Results from time and probability analysis)

Acknowledgements The authors gratefully acknowledge the financial assistance provided by the National Natural Science Foundation of China and the Natural Science Foundation of Jiangsu Province.

Author Contributions Hang Shen, Xiang Li, and Yan Wang wrote the main manuscript text. Tianjing Wang and Guangwei Bai provided guiding ideas and suggestions. All authors reviewed the manuscript.

Funding This work was supported in part by the National Natural Science Foundation of China under Grants 61502230 and 61501224, the Natural Science Foundation of Jiangsu Province under Grant BK20201357, and the Six Talent Peaks Project in Jiangsu Province under Grant RJFW-020.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing Interests The authors declare no competing interests.

Ethical Approval and Consent to Participate This article does not contain any studies with human participants or animals performed by any of the authors.

Consent for Publication All authors agree to publish the paper and related research results of the paper.

References

1. Lilhore UK, Dalal S, Simaiya S (2024) A cognitive security framework for detecting intrusions in IoT and 5G utilizing deep learning. Comput Sec 136:103560
2. Shen H, Ye Q, Zhuang W, Shi W, Bai G, Yang G (2021) Drone-small-cell-assisted resource slicing for 5G uplink radio access networks. IEEE Trans Veh Technol 70(7):7071–7086
3. Li F, Shen H, Mai J, Wang T, Dai Y, Miao X (2024) Pre-trained language model-enhanced conditional generative adversarial networks for intrusion detection. Peer-to-Peer Netw Appl 17(1):227–245
4. Stellios I, Kotzanikolaou P, Psarakis M, Alcaraz C, Lopez J (2018) A survey of IoT-enabled cyberattacks: Assessing attack paths to critical infrastructures and services. IEEE Comm Surv Tutor 20(4):3453–3495
5. Sun X, Dai J, Liu P, Singhal A, Yen J (2018) Using Bayesian networks for probabilistic identification of zero-day attack paths. IEEE Trans Inf Forensics Secur 13(10):2506–2521
6. Jayasuryapal G, Pranay PM, Kaur H et al (2021) A survey on network penetration testing. In international conference on intelligent engineering and management (ICIEM), pp 373–378
7. Hu H, Liu Y, Zhang H, Yang Y, Ye R (2018) Route prediction method for network intrusion using absorbing markov chain. J Comput Res Dev 55(4):831–845
8. Kotenko I, Chechulin A (2012) Attack modeling and security evaluation in siem systems. Int Trans Syst Sci Appl 8:129–147
9. Fang X, Zhai L, Jia Z, Bai W (2014) A game model for predicting the attack path of apt. In IEEE 12th international conference on dependable, autonomic and secure computing, pp 491–495
10. Shen H, Tian Y, Wang T, Bai G (2023) Slicing-based task offloading in space-air-ground integrated vehicular networks. IEEE Trans Mob Comput 23(5):4009–4024
11. Uprety A, Rawat DB (2020) Reinforcement learning for IoT security: A comprehensive survey. IEEE Internet Things J 8(11):8693–8706
12. Wang B, Liu Z, Li Q, Prorok A (2020) Mobile robot path planning in dynamic environments through globally guided reinforcement learning. IEEE Robot Autom Lett 5(4):6932–6939
13. Cody T, Rahman A, Redino C, Huang L, Clark R, Kakkar A, Kushwaha D, Park P, Beling P, Bowen E (2022) Discovering exfiltration paths using reinforcement learning with attack graphs. In IEEE conference on dependable and secure computing (DSC), pp 1–8
14. Zhang L, Li H, Shiming Xia Y, Pan WB, Feng Q, Li Wei, Zheng Qibin, Guo Shize, Pan Zhisong (2022) Discover the hidden attack path in multiple domain cyberspace based on reinforcement learning. Sci Program 2022(1):6008447
15. Hu Z, Beuran R, Tan Y (2020) Automated penetration testing using deep reinforcement learning. In IEEE European symposium on security and privacy workshops (EuroS&PW), p 2–10
16. Somesula MK, Rout RR, Somayajulu DVLN (2022) Cooperative cache update using multi-agent recurrent deep reinforcement learning for mobile edge networks. Comput Netw 209:108876
17. Hao J, Yang T, Tang H, Bai C, Liu J, Meng Z, Liu P, Wang Z (2024) Exploration in deep reinforcement learning: From single-agent to multiagent domain. IEEE Trans Neural Netw LearnSyst 35(7):8762–8782
18. Li X, Huangfu W, Xinyi X, Huo J, Long K (2024) Secure offloading with adversarial multi-agent reinforcement learning against intelligent eavesdroppers in uav-enabled mobile edge computing. IEEE Trans Mob Comput 23(12):13914–13928
19. Ju Y, Chen Y, Cao Z, Liu L, Pei Q, Xiao M, Ota K, Dong M, Leung VCM (2023) Joint secure offloading and resource allocation for vehicular edge computing network: A multi-agent deep reinforcement learning approach. IEEE Trans Intell Transp Syst 24(5):5555–5569, 2023
20. Liu X, Zhang H, Zhang Y, Shao L (2020) Optimal network defense strategy selection method based on evolutionary network game. Secur Commun Netw 2020(1):5381495
21. Khan AA, Wagan AA, Laghari AA, Gilal AR, Aziz IA, Talpur BA (2022) Biomt: A state-of-the-art consortium serverless network architecture for healthcare system using blockchain smart contracts. IEEE Access 10:78887–78898
22. Atzmon D, Zax Y, Kivity E, Avitan L, Morag J, Felner A (2020) Generalizing multi-agent path finding for heterogeneous agents. Proc Int Symp Comb Search 11:101–105
23. Sharma PK, Fernandez R, Zaroukian E, Dorothy M, Basak A, Asher DE (2021) Survey of recent multi-agent reinforcement learning algorithms utilizing centralized training. In artificial intelligence and machine learning for multi-domain operations applications III, vol 11746. pp 665–676
24. Rawat DB, Njilla L, Kwiat K, Kamhoua C (2018) iShare: Blockchain-based privacy-aware multi-agent information sharing games for cybersecurity. In international conference on computing, networking and communications (ICNC), pp 425–431
25. Cui J, Liu Y, Nallanathan A (2019) Multi-agent reinforcement learning-based resource allocation for UAV networks. IEEE Trans Wireless Commun 19(2):729–743
26. Bostock RM (1999) Signal conflicts and synergies in induced resistance to multiple attackers. Physiol Mol Plant Pathol 55(2):99–109
27. Hausken K, Bier VM (2011) Defending against multiple different attackers. Eur J Oper Res 211(2):370–384
28. Bertenyi B, Burbidge R, Masini G, Sirotkin S, Gao Y (2018) Ng radio access network (ng-ran). J ICT Stand 6(1–2):59–76
29. Khichane A, Fajjari I, Aitsaadi N, Gueroui M (2023) 5GC-Observer: a non-intrusive observability framework for cloud native 5G system. In IEEE/IFIP network operations and management symposium (NOMS), pp 1–10
30. Lin CS (2022) A lightweight design of 5g private network with edge computing. In Int Conf Mechatronics Technol (ICMT), pp 1–2

31. Ikeda T, Shibuya T (2022) Centralized training with decentralized execution reinforcement learning for cooperative multi-agent systems with communication delay. In annual conference of the society of instrument and control engineers (SICE), pp 135–140
32. Branzei R, Dimitrov D, Tijs S (2008) Models in cooperative game theory, vol 556. Springer Science & Business Media
33. Shapley LS (1953) A value for n-person games. Contribution to the Theory of Games, 2
34. Nash JF Jr (1950) Equilibrium points in n-person games. Proc Natl Academy Sci 36(1):48–49
35. Wan K, Dingwei W, Li B, Gao X, Zijian H, Chen D (2022) ME-MADDPG: An efficient learning-based motion planning method for multiple agents in complex environments. Int J Intell Syst 37(3):2393–2427
36. Factored multi-agent centralised policy gradients (2021) Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac. Adv Neural Inf Process Syst 34:12208–12221
37. Chu T, Wang J, Codecà L, Li Z (2019) Multi-agent deep reinforcement learning for large-scale traffic signal control. IEEE Trans Intell Transp Syst 21(3):1086–1095
38. Shi H, Tian Y, Li H, Huang J, Shi L, Zhou Y (2024) Task offloading and trajectory scheduling for UAV-enabled MEC networks: An madrl algorithm with prioritized experience replay. Ad Hoc Netw 154

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Hang Shen received the Ph.D. degree with honors in Computer Science from the Nanjing University of Science and Technology, Nanjing, China, in 2015. He worked as a Full-Time Postdoctoral Fellow with the Broadband Communications Research (BBCR) Lab, Electrical and Computer Engineering Department, University of Waterloo, Waterloo, ON, Canada, from 2018 to 2019. He is an Associate Professor with the Department of Computer Science and Technology at Nanjing Tech University, Nanjing, China. His research interests involve artificial intelligence for cybersecurity and wireless networking. He serves as an Associate Editor for *Journal of Information Processing Systems*, *Frontiers in Blockchain*, and IEEE Access. He serves/served as a TPC Member of the 2024 IEEE International Conference on High Performance Computing and Communications (HPCC) and the 2021 Annual International Conference on Privacy, Security and Trust (PST). He is an Executive Committee Member of the ACM Nanjing Chapter, an IEEE member, and a CCF Senior Member.

Xiang Li received the B.Eng. degree in Network Engineering from Nanjing Institute of Technology, Nanjing, China. He is currently working toward the M.Eng. degree in Computer Science with Nanjing Tech University, Nanjing, China. His research interests include large language models and knowledge graphs for cybersecurity. He is a CCF Student Member.

Yan Wang received the B.Eng. and M.Eng. degrees both in Computer Science from Nanjing Tech University, Nanjing, China, in 2021 and 2024. His research interests include multi-agent deep reinforcement learning and large language models for cybersecurity.

Tianjing Wang holds a B.Sc. in Mathematics from Nanjing Normal University in 2000, an M.Sc. in Mathematics from Nanjing University in 2002, and a Ph.D. in Signal and Information System from the Nanjing University of Posts and Telecommunications (NUPT) in 2009, all in Nanjing, China. From 2011 to 2013, she was a Full-Time Postdoctoral Fellow with the School of Electronic Science and Engineering at NUPT. From 2013 to 2014, she was a Visiting Scholar with the Electrical and Computer Engineering Department at the State University of New York at Stony Brook, NY, USA. She is an Associate Professor in the Communication Engineering Department at Nanjing Tech University, Nanjing, China. Her research interests include cybersecurity and V2X communication networks. She is a member of IEEE and CCF.

Guangwei Bai received the B.Eng. and M.Eng. degrees in computer engineering from Xi'an Jiaotong University, Xi'an, China, in 1983 and 1986, respectively, and the Ph.D. degree in Computer Science from the University of Hamburg, Hamburg, Germany, in 1999. From 1999 to 2001, he worked as a Research Scientist at the German National Research Center for Information Technology, Germany. In 2001, he joined the University of Calgary, Calgary, AB, Canada, as a Research Associate. Since 2005, he has been working at Nanjing Tech University, Nanjing, China, as a Professor in Computer Science. From October to December 2010, he was a Visiting Professor with the Electrical & Computer Engineering Department at the University of Waterloo, Waterloo, ON, Canada. His research interests include architecture and protocol design for future networks, cybersecurity, and privacy computing. He has authored and co-authored more than 80 peer-reviewed papers in international journals and conferences.