Automated Penetration Testing with Fine-Grained Control through Deep Reinforcement Learning

Xiaotong Guo, Jing Ren, Jiangong Zheng, Jianxin Liao, Chao Sun, Hongxi Zhu,
Tongyu Song, Sheng Wang, Wei Wang

***Abstract*—Penetration testing (PT) is an active method of evaluating the security of a network by simulating various types of cyber attacks in order to identify and exploit vulnerabilities. Traditional PT involves a time-consuming and labor-intensive process that is prone to errors and cannot be easily formulated. Researchers have been investigating the potential of deep reinforcement learning (DRL) to develop automated PT (APT) tools. However, using DRL in APT is challenged by partial observability of the environment and the intractability problem of the huge action space. This paper introduces RLAPT, a novel DRL approach that directly overcomes these challenges and enables intelligent automation of the PT process with precise control. The proposed method exhibits superior efficiency, stability, and scalability in finding the optimal attacking policy on the simulated experiment scenario.**

***Keywords*—APT, cyber attack, DRL**

I. Introduction

Penetration testing (PT) plays a critical role in evaluating the security of a computer network, which aims to identify and exploit security vulnerabilities in the network by manually simulating malicious attacks. Whereas the PT process could be time-consuming and labor-intensive, especially for large and complex networks. Additionally, the test results are primarily dependent on the experience and knowledge of the human penetration testers. Therefore, the growing demand for frequent, efficient, and cost-effective PT gives rise to the proposal of automated penetration testing (APT)[1].

Utilizing artificial intelligence (AI) for APT is an emerging line of research that has been considered by several research works, desirable for emulating human experts with an attacking strategy instead of naively testing all possible attacks. Early works model the PT process as attack graphs[2-4], recognizing the PT process as a sequential decision making process. However, these approaches require complete knowledge of the network topology and host configurations, which is unrealistic in practical PT scenarios.

Recently, applying deep reinforcement learning (DRL) to PT is gaining prevalence in academia[5-8]. As a paradigm of learning in which the autonomous agent interacts with the environment and learns through trial and error to find out an optimal acting policy, DRL has achieved human-level performance on various applications and platforms[9-11], and appears to be a suitable candidate for tackling the APT problem. Like a human penetration tester, the DRL agent can be trained to launch emulated attacks based on partial observation of the target network. The attained network information gradually increases when the DRL agent discovers and infects new network nodes.

To efficiently train the DRL agent to take optimal automated actions to emulate cyber attacks, a dynamic simulated training environment with appropriate feedback for the actions taken is essential. Simulation platforms for DRL-based cyber battle research have been proposed to facilitate the training, including CybORG[12], CyberBattleSim[13], and CyGIL[14]. The simulation environments mentioned above have been widely used in the existing research[15-18].

Nevertheless, applying DRL in APT encounters some difficulties. The partial observability issue poses a significant hurdle in the APT process as the agent is unable to access comprehensive information about the target system beforehand. Consequently, the agent must progressively enhance its understanding by iteratively gathering information throughout the process. The performance of vanilla DRL methods (e.g., deep Q-network (DQN)[19]) could suffer from partial observ-

Manuscript received Jul. 21, 2023; revised Aug. 27, 2023; accepted Sep. 11, 2023. This work was supported by the National Key R&D Program of China under Grant 2020YFB1807503 and the National Natural Science Foundation of China under Grant U20A20156, Grant 62001087, and Grant 62201309. (Xiaotong Guo and Jing Ren contribute equally in this work.) The associate editor coordinating the review of this paper and approving it for publication was W. Zhang.

X. T. Guo, J. Ren, J. G. Zheng, J. X. Liao, C. Sun, H. X. Zhu, T. Y. Song, S. Wang. University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: 202121011514@std.uestc.edu.cn; renjing@uestc.edu.cn; zjg@std.uestc.edu.cn, 202121010710@std.uestc.edu.cn, 202121010701@std.uestc.edu.cn, 202221011301@std.uestc.edu.cn; tongyusong@ieee.org; wsh_keylab@uestc.edu.cn).

J. Ren, W. Wang. Peng Cheng Laboratory, Shenzhen 518000, China (e-mail: renj@pcl.ac.cn; wangw01@pcl.ac.cn).

ability since no explicit mechanisms for interpreting the underlying state of the environment are incorporated[20]. These approaches are only effective when the observations are reflective of the underlying system states. Schwartz et al.[5] and Hu et al.[6] utilized the partially observable Markov decision process (POMDP)[21] to model the APT process, and employed conventional reinforcement learning (RL) or DRL method to learn exploit strategies. These previous studies have demonstrated that RL or DRL could be effective when handling small-scale networks, which have relatively small action spaces. However, extending DRL to networks with a larger action space remains a challenge[5].

Some efforts were made to decompose the huge action space of APT process. Tran et al.[7] proposed a novel DRL architecture with hierarchically structured agents. The algebraic action decomposition method was employed in this approach to split the large action space into manageable sets for separate DQN agents. Despite this, there remains significant room for optimization within the hierarchical architecture at the algorithmic level[8].

In summary, APT tools leveraging DRL face various challenges, including a) environment uncertainties due to partial observability of the target network, and b) intractability problem of a huge, discrete attack action space which scales exponentially with the complexity of the target network.

In this paper, we frame the APT process as a POMDP with partial knowledge of the network and a huge action space, and propose a novel DRL-based method called RLAPT to automate the PT process efficiently and accurately. We conduct extensive experiments on CyberBattleSim to evaluate the validity of our method. To the authors' best knowledge, this work is the first to directly address all the aforementioned challenges in applying DRL to APT. The key contributions of this work are as follows.

- We construct a well-crafted observation space for the APT agent, which enables it to gain knowledge of the system gradually through exploration and exploitation, and thus make more accurate decisions.
- We decompose the action space into multiple sub-action spaces to handle the intractability problem of the agent's huge action space. The promoted action decomposition strategy achieves dimensionality reduction of the agent's huge action space, and enables the agent to master fine-grained control of the attack behavior, which stabilizes the agent's learning and enhances the convergence performance.
- We develop various optimization strategies for APT to boost the efficiency and accuracy of the agent, including action mask for exploration efficiency and probability mask for stable training.
- Experimental results show that the trained AI agent can master fine-grained control of the PT process, and achieves superior and stable convergence performance.

The rest of the paper is structured as follows. A brief review of CyberBattleSim is given in section II. Section III discusses the problem formulation of the APT process. Section IV gives specific details of RLAPT. Experimental results, including a performance comparison of the different APT methods and an analysis of parameter configurations are presented in section V. Finally, section VI summarizes our conclusions and provides directions for the future.

II. Brief Overview of CyberBattleSim

Training AI on simulation platforms rather than in the real environment possesses multiple advantages, among which are cost-effectiveness, safety, speed, repeatability, scalability, and etc. CyberBattleSim provides a high-level abstraction of computer networks and cyber security concepts. It enables researchers to create a virtual network with a specific parameterized structure and a set of predefined vulnerability libraries that the DRL-based APT agent (attacker) can exploit to laterally move through the network.

In any simulated network on CyberBattleSim platform, there will be some nodes running various operating systems and software. Each node has a set of properties (including the operating system, running services, and potential existing risks), a node value, and some pre-assigned vulnerabilities. However, the environment is partially observable to the attacker, which means that the attacker cannot see all nodes and network information in advance. It is assumed that only one node in the network is initially discovered and infected (or owned) by the attacker. From the attacker's perspective, each node in the network can be in one of three states: undiscovered, discovered but not yet owned, or owned. The attacker tries to explore different types of exploits on vulnerabilities in the network, and maximize the cumulative long-term reward earned from successful attacks. The attacker's goal is to take ownership of some portion of the network through sustained attack behaviors.

There exist three different types of actions for attack in CyberBattleSim: performing a local attack, performing a remote attack, and connecting to some other node. Local attacks are executable only on owned nodes, targeted at the local vulnerabilities. Remote attacks are executable on any discovered node, targeted at the remote vulnerabilities. Connect exploits are executable on any discovered node as well, targeted at node's ports, and enable the attacker to take ownership of the target node using leaked credentials. A successfully launched attack could yield various outcomes, such as discovering new nodes, obtaining leaked credentials, obtaining node properties, taking ownership of the target node, or escalating privilege on the target node. The attacker is equipped with a local vulnerability library, a remote vulnerability library, and a port

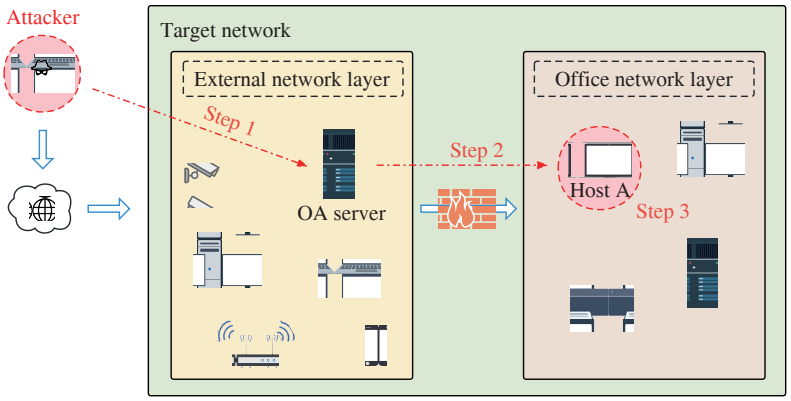


Fig. 1 An example of the APT process in a target network

library, where the target local vulnerability, remote vulnerability, and port can be chosen, respectively.

III. Problem Formulation

Fig. 1 depicts a toy example of the APT process in a target network. The target network consists of an external network layer and an office network layer. The attacker can establish direct communication with nodes on the external network layer, but cannot access nodes on the office network layer. Nodes on the office network layer can communicate with nodes on the external network layer. As shown in Fig. 1, the simulated attacker manages to break into the target network and take ownership of Host A. Initially, the attacker exploits the remote vulnerability CVE-2021-49104 to own the office automation (OA) server. Then the leaked credentials are leveraged to sign into Host A. Finally, the attacker scans Host A's home directory and discovers its new properties.

We model the APT process as a POMDP. Formally, a POMDP can be represented with a tuple $\langle S, A, T, R, \Omega, O \rangle$. S is a finite set of states of the target network for penetration testing, and Ω is a finite set of observations that the attacker can experience of the target network. A is a finite set of attack actions available to the attacker. At each time t, the attacker is in some system state $s_t \in S$ and perceives an observation $o_t \in \Omega$. The observation o_t perceived depends on the probability distribution $o_t \sim O(s_t)$. The attacker therefore executes an attack $a_t \in A$, and transfers to the next state s_{t+1} according to $T(s_t, a_t, s_{t+1})$, which is a conditional probability function denoted by $P(s_{t+1}|s_t, a_t)$. $R : S \times A \to \mathbb{R}$ is the state-action dependent reward function, giving the immediate reward signal $r_t \sim R(s_t, a_t)$ gained by taking each action in each state.

The solution to a POMDP problem is an optimal policy π^* that maps observations to actions so as to maximise the expected total reward $G_t = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau$, where $\gamma \in [0, 1]$ is the discount factor that determines the relative importance of the long-term rewards.

IV. RLAPT Architecture

The following section describes how our novel DRL-based APT method RLAPT solves: 1) the challenge of partial observability in APT, and 2) the intractability problem of the agent's action space. Our proposed method adopts the actor-critic framework[22], and utilizes proximal policy optimization (PPO)[23] algorithm to guide the agent's learning process. Several strategies are incorporated into our method to train the agent more effectively and efficiently. Fig. 2 gives a detailed illustration of RLAPT, and the training procedure of RLAPT is described in Algorithm 1. The details of our method will be further explained in this section.

A. Observation Space Design

Since the information gathered by the attacker is only partial glimpses of the underlying tested network system state, an effective design for the observation space is required so as to enable the attacker to perceive the environment more effectively, and furnish the attacker with a more perspicuous learning direction. It is worth noting that the attacker discovers new network nodes and other critical information by consecutively exploring and successfully exploiting network vulnerabilities, so the observation space should be designed in an incremental manner.

However, DRL approaches rely on input observations with a fixed dimensionality. Therefore, our method has a capacity limit when handling the varying discovered nodes. To deal with this limitation, we set the maximum number of the discovered nodes that RLAPT can deal with as $N_{\max}$. Specifically, the observation $o_t \in \Omega$ consists of the following features.

$$o_t = \{n_{\text{dis}}(t), n_{\text{own}}(t), L(t), n_{\text{port}}(t), n_{\text{cred}}(t), P(t)\}, \quad (1)$$

$n_{\text{dis}}(t)$ is the number of discovered nodes at time t, and $n_{\text{own}}(t)$ is the number of owned nodes at time t. $L(t)$ is a binary vector denoting whether each discovered node is owned or not yet.

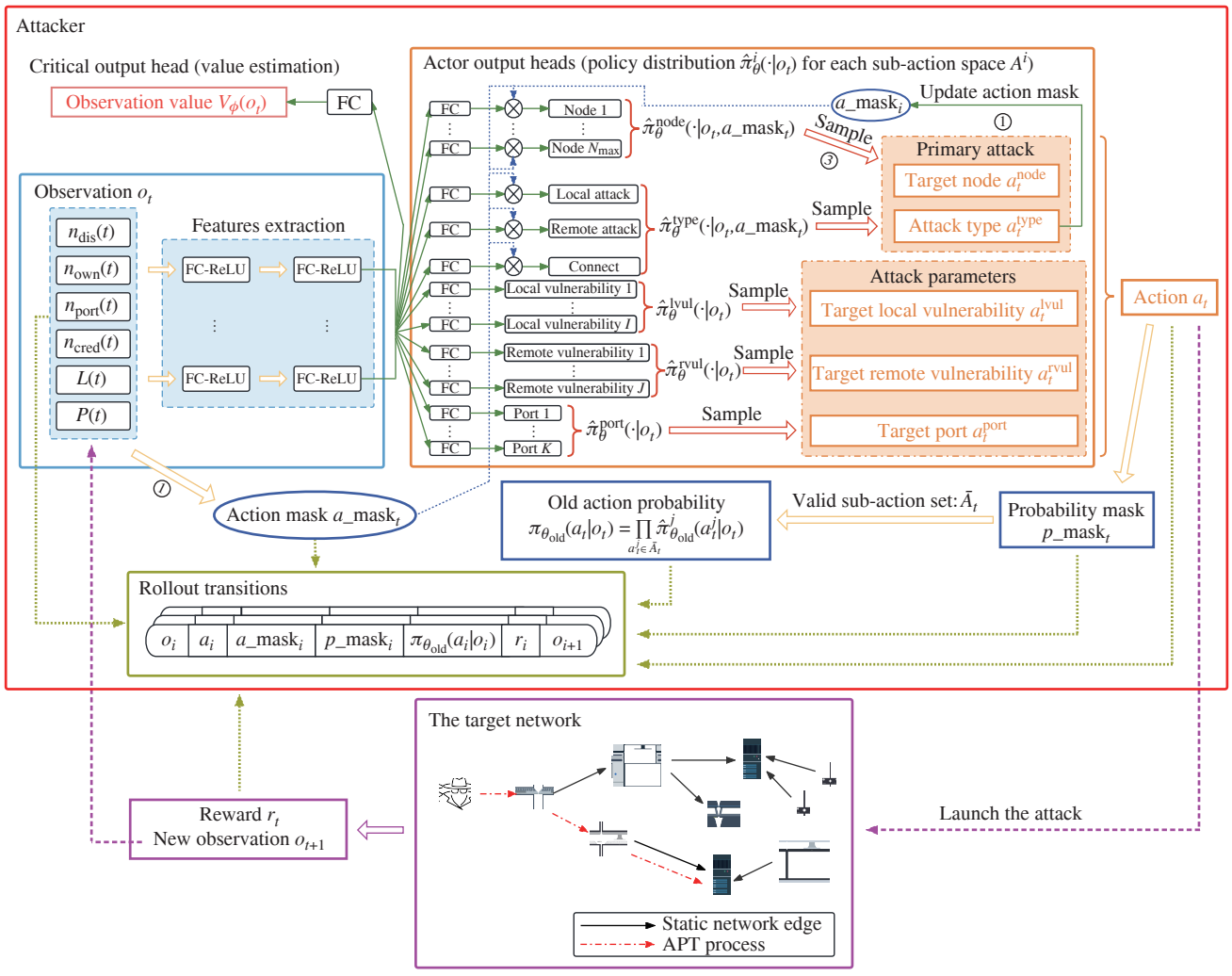


Fig. 2 Illustrations of our promoted automated penetration testing method RLAPT

$n_{port}(t)$ denotes the number of leaked ports on each discovered node, and $n_{cred}(t)$ denotes the number of leaked credentials on each discovered node. The length of $L(t)$, $n_{port}(t)$ and $n_{cred}(t)$ is N_{max}. $P(t)$ is also a binary vector and represents the extracted node properties of each discovered node (1 for discovered properties, 0 for unclear or unpossessed properties). Considering N_{prop} different node properties, the length of $P(t)$ should be $N_{max} \cdot N_{prop}$.

Each of the mentioned features is arranged in a specific manner according to the nodes' discovery order. The ith value of $n_{port}(t)$, $n_{cred}(t)$ and $L(t)$ imply the corresponding feature value of the ith discovered nodes, and the ith N_{prop} values of $P(t)$ represents the extracted properties of the ith discovered nodes. In this way, the attacker is able to gradually improve its comprehension of the target network, and thus make more accurate and effective attack decisions. When fewer than N_{max} nodes are discovered ($n_{dis}(t) < N_{max}$), the redundant parts of each observation feature are filled with zeros. Feature normalization is conducted on each dimension of the observation space for effective training.

B. Multi-Discrete Action Space Design

As mentioned in section I, it is difficult and unstable to apply conventional DRL algorithms to automate the PT process as the action space could explode to be extremely large, even in relatively small networks. We opt for the multi-discrete action space design to handle the intractability problem of the huge action space, which decomposes the action space into smaller and manageable subsets, and lets the attacker produce each sub-action with different output heads. Therefore, each sub-action space has its own policy distribution and output heads, and contributes equally to get the reward signal. Additionally, each sub-action within a complete attack action is treated independently during the training process to decouple their inter-correlations. This action decomposition scheme has been proven to be effective in dealing with the large action

space on various platforms[24-26].

To be specific, in RLAPT the attack action space is decomposed into several different sub-action spaces as target node space A_{node}, attack type space A_{type}, target local vulnerability space A_{lvul}, target remote vulnerability space A_{rvul}, and target port space A_{port}. An attack action comprises both a designated target and the method or tools used for the attack. Hence, we partition the entire attack action into two parts: primary attack and attack parameters.

The target node and attack type make up the primary attack, indicating which node to attack and the certain attack type. The dimension of A_{node} is set to be $N_{\max}$. It is worth mentioning that the number of valid target nodes gradually increases as the number of discovered nodes increases through the successful attacks. A_{type} includes three kinds of attack actions following the setting of CyberBattleSim: local attack, remote attack, and connect. Note that A_{type} can be easily extended to other settings.

A_{lvul}, A_{rvul}, and A_{port} are attack parameter spaces. Different types of attacks have different valid attack parameters. The valid attack parameters for local attack, remote attack, and connect are local vulnerability, remote vulnerability, and port, respectively. The attack parameters are picked from the attacker's vulnerability and port libraries. The dimension of each attack parameter space is aligned with the size of the related library.

Given any perceived observation o_t, the attacker will get the complete action as the combination of the output five sub-actions as $a_t = (a_{\text{node}}(t), a_{\text{type}}(t), a_{\text{lvulid}}(t), a_{\text{rvulid}}(t), a_{\text{pid}}(t))$. Based on the sampled primary attack and valid attack parameter, the attacker is capable of initiating a practical attack with fine-grained control, such as attempting to exploit a potential local vulnerability on an owned node. Irrelevant parameter actions are simply ignored. Notably, connect exploits require specifying the credential to be used. If the attacker has obtained the credential for the target port of the target node, the leaked credential will be used in the attack. Otherwise, if the target node has any leaked credentials, one of them will be randomly selected for use in the attack. If neither of these conditions is met, a randomly selected leaked credential from the cache will be used.

C. Reward Function Design

Generally, the attacker will get positive feedback when the launched attack is valid, or get punished with a negative reward signal when the previous attack fails. The specific reward value is determined by the attack outcome. In RLAPT, we design our reward function (as shown in Tab. 1) referring to the reward function design from CyberBattleSim, which considers 14 different attack results[13]. It is worth mentioning that the reward function design could be easily adapted to account for more attack situations or various attack purposes.

Tab. 1 Reward design

Attack outcome	Reward value
Successfully launching an attack	7
Discovering a new node	5
Discovering a new credential	3
Discovering a new node property	2
Attack failure	−1

Algorithm 1 Training procedure of RLAPT

Randomly initialize the critic network $V_\phi(o)$ and actor network $\pi_\theta(o, a\_\text{mask})$ with weights ϕ and θ
Initialize replay buffer R
for episode = 0 to N **do**
  Receive the initial observation o_0 of the tested system
  for $t = 0$ to T **do**
    if the attack goal is realized **then**
      break
    end if
    Update the action mask a_mask_t based on o_t to marginalize out invalid attacks
    Select an action a_t according to the current policy $\pi_\theta(o_t, a\_\text{mask}_t)$
    Update the probability mask p_mask_t for action a_t to ignore the irrelevant sub-action probabilities
    Compute the current action probability $prob_t$
    Execute action a_t and observe reward r_t and obtain new observation o_{t+1}, and get the transition $(o_t, a_t, a\_\text{mask}_t, p\_\text{mask}_t, prob_t, r_t, o_{t+1})$
    Store this transition in replay buffer R
  end for
  Compute advantage estimates $\hat{A}_i$ for each transition $(o_i, a_i, a\_\text{mask}_i, p\_\text{mask}_i, prob_i, r_i, o_{i+1})$ in replay buffer R using GAE
  Optimize objective function J_θ and MSE loss L_ϕ, with K epochs and minibatch size M
  Refresh the replay buffer R
end for

D. Training Logic

The state-of-the-art model-free DRL algorithm PPO is utilized to guide the attacker's learning process effectively and efficiently. PPO has an actor-critic architecture. The actor neural network is responsible for selecting actions in the environment, while the critic neural network estimates the value function. The objective function of PPO for the actor network is defined as follows.

$$J_\theta = \hat{E}_t\Big[\min\Big(\text{clip}(m_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t, m_t(\theta)\hat{A}_t\Big) + \alpha H(\pi_\theta(\cdot|o_t))\Big], \tag{2}$$

where $m_t(\theta) = \frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{\text{old}}}(a_t|o_t)}$ is the probability ratio between old policy and new policy, and $\alpha H(\pi_\theta(\cdot|o_t))$ is the policy entropy term. Generalized advantage estimator (GAE)[27] is used to estimate the advantage function $\hat{A}_t$

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V , \tag{3}$$

where γ and λ are discount parameters for the bias-variance trade-off when using the approximate value function, and $\delta^V_{t+l} = r_{t+l} + \gamma V_\phi(o_{t+l+1}) - V_\phi(o_{t+l})$ can be considered as an estimate of the advantage of the action a_t. The loss function for the critic network is the mean square error (MSE) between the estimated and target values

$$L_\phi = \frac{1}{N}\sum_{i=1}^{N} \left\| V_\phi(o_i) - \hat{V}_i \right\|^2, \tag{4}$$

where $\hat{V}_t = \sum_{l=0}^{n-1} \gamma^l r_{t+l} + \gamma^n V_\phi(o_{t+n})$ is the n-step estimation of the value function[28]. The hidden layers of the actor and critic network are fully connected, and utilize rectified linear unit (ReLU) as the activation function.

We develop various optimization strategies for RLAPT to enhance the performance of the attacker, and two key components of RLAPT are mentioned in the forthcoming text.

1) Action Mask for Invalid Primary Attacks: One effective approach to facilitate the training process is to marginalize out invalid attacks with the action masks. Invalid attacks refer to those that are guaranteed to fail, including local attacks on unowned nodes and connect exploits with no leaked credentials. When fewer than $N_{\max}$ nodes are discovered, the redundant part of the target node space should be masked out as well.

Therefore, when initiating an attack, the attacker follows the procedures outlined below to determine the primary attack: a) generating an action mask based on the current observation (① in Fig. 2); b) choosing a valid attack type and then updating the action mask on target nodes (② in Fig. 2); c) selecting a valid target node (③ in Fig. 2). During the training phase, the action mask is also applied when calculating the action probability and policy entropy. This results in a new set of probability distributions for the primary attack where the probability of any invalid primary attack is set to zero.

For illustration purposes, consider the situation when the attacker only discovers three nodes in the target network system, with no nodes conquered successfully yet. Let $N_{\max} = 5$. Given the parameterized policies

$$\pi^\theta_{\text{node}}(\cdot\,|\,o) = [0.1, 0.1, 0.3, 0.4, 0.1], \tag{5a}$$
$$\pi^\theta_{\text{type}}(\cdot\,|\,o) = [0.3, 0.6, 0.1], \tag{5b}$$

where π^θ_{node} determines the target node and π^θ_{type} determines the launched attack type. The attacker generates the action mask for each primary attack given the observation:

$$a\_\text{mask}_{\text{node}} = [1, 1, 1, 0, 0], \tag{6a}$$
$$a\_\text{mask}_{\text{type}} = [0, 1, 0]. \tag{6b}$$

Since the attacker discovers three nodes, the last two elements of $a\_\text{mask}_{\text{node}}$ should be 0. Only the remote attack is set to be valid as no nodes have been conquered so far. Therefore we can get a new set of probability distributions (after renormalization), from which the primary attack sub-actions are sampled:

$$\pi^\theta_{\text{node}}(\cdot\,|\,o, a\_\text{mask}_{\text{node}}) = Norm(\pi^\theta_{\text{node}}(\cdot\,|\,o) \odot a\_\text{mask}_{\text{node}}) = [0.2, 0.2, 0.6, 0, 0], \tag{7a}$$
$$\pi^\theta_{\text{type}}(\cdot\,|\,o, a\_\text{mask}_{\text{type}}) = Norm(\pi^\theta_{\text{type}}(\cdot\,|\,o) \odot a\_\text{mask}_{\text{type}}) = [0.0, 1.0, 0.0]. \tag{7b}$$

Notably, $\pi^\theta_{\text{node}}(\cdot\,|\,o, a\_\text{mask}_{\text{node}})$ and $\pi^\theta_{\text{type}}(\cdot\,|\,o, a\_\text{mask}_{\text{type}})$ above are real probability distributions, from which sub-actions are sampled from. A renormalization operation should be conducted right after the masking process to yield real probability distributions for decision-making.

2) Probability Mask for Irrelevant Attack Parameters: For any valid attack, some attack parameters are irrelevant depending on the type of the primary attack (e.g., target remote vulnerability and target port sub-actions are irrelevant for any local attack). If these irrelevant attack parameters are taken into account when calculating the action probability or policy entropy during the training process, the noise will be incorporated into the policy gradients, which hinders the attacker's training.

Hence, we generate the probability mask $p\_\text{mask}(t)$ for the irrelevant attack parameters to get the valid sub-action set $\bar{A}_t$ of action a_t (e.g., $\bar{A}_t = (a_{\text{node}}(t), a_{\text{type}}(t), a_{\text{vulid}}(t))$ when a_t is a local attack). We utilize the probability mask to mask out the probabilities and policy entropies of irrelevant attack parameters when optimizing. Then the action probability $\pi_\theta(a_t|o_t)$ in (2) is the product of all the relevant sub-action probabilities of action a_t

$$\pi_\theta(a_t|o_t) = \prod_{a_i \in \bar{A}_t} \hat{\pi}^i_\theta(a_i|o_t)\,, \tag{8}$$

where $\hat{\pi}^i_\theta(\cdot\,|o_t)$ is the policy distribution for the a_i-related sub-action space. The old action probability $\pi_{\theta_{\text{old}}}(a_t|o_t)$ is calculated in the similar way. The policy entropy $H(\pi_\theta(\cdot\,|o_t))$ in (2) transforms into the sum of the policy entropy of each relevant sub-action space

$$H(\pi_\theta(\cdot\,|o_t)) = \sum_{a_i \in \bar{A}_t} H(\hat{\pi}^i_\theta(\cdot\,|o_t))\,, \tag{9}$$

where $H(\hat{\pi}^i_\theta(\cdot\,|o_t))$ is the policy entropy of the related policy distribution. This trick can significantly improve the exploration efficiency and performance of the attacker. The procedure of attack parameter masking process is similar to that of primary action masking process.

V. Experimental Results

We test our method using the capture-the-flag (CTF) simulated scenario from CyberBattleSim[13]. A comprehensive

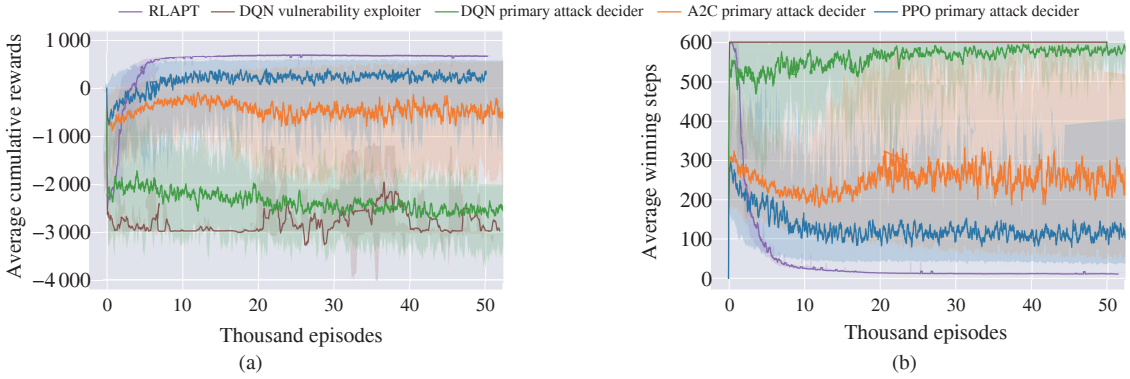


Fig. 3 Results of the RLAPT agent and comparing agents on CTF simulated scenario. The agents are tested every 10 training episodes. Fig. 3(a) shows the cumulative rewards, and Fig. 3(b) shows the winning steps. If the agent fails to reach the attack goal, the winning step is set to be the maximum time step. It takes approximately 4 147 steps on average for the random attacker to attain the attack goal, resulting in an average accumulated reward of roughly −20 938. Curves are smoothed with a moving average over 20 points: (a) Cumulative reward; (b) Winning steps

Tab. 2 Environmental settings of Capture-The-Flag scenario

Environment parameter	Value
Nodes count	10
Maximum number of nodes seized	6
Pre-assigned local vulnerability types	3
Pre-assigned remote vulnerability types	8
Running ports count	7
Incorporation of defender	False

description of the tested scenario is given in Tab. 2. We account for 10 different node properties, and set $N_{\max}$ to be 10. All together this produces a combined factorized action space size of up to $10 \times 3 \times 3 \times 8 \times 7 = 5\,040$ dimensions. The attacker achieves its attack goal only after it has occupied all the nodes that can be seized in the system. We compare the following four distinct categories of agents.

- **Random Attacker:** Attacker that randomly launches effective attacks.
- **Multi-agent reinforcement learning (MARL)-based Attacker:** Attacker that employs hierarchically structured reinforcement learning agents to handle the decomposed large action space, where each agent is responsible for the control of some certain sub-action[7].
- **DRL-based Vulnerability Exploiter:** DRL-based attacker promoted by Microsoft in CyberBattleSim, which outputs the target vulnerability or port for each attack. The target node and relevant attack parameters are randomly selected from valid sets. A different observation space design is adopted. DQN Vulnerability Exploiter is implemented by Microsoft.
- **DRL-based Primary Attack Deciders:** DRL-based attackers that determine the primary attack (i.e., target node and attack type). The relevant attack parameters are randomly selected from the predefined libraries. DQN, A2C[22], and PPO primary attack deciders are implemented.

The last two types of agents leverage the same logic for selecting credentials as RLAPT. Each DRL-based agent in our experiment utilizes its respective classical neural network architecture and parameter configurations.

We evaluate the performance of each DRL-based agent by running extensive simulations on the CTF scenario following the same experimental settings. We run a total of 30 000 episodes for each experiment with each episode running for a maximum of 600 time steps. Each experiment is repeated five times with different random seeds to ensure reproducibility. The random attacker is tested for 30 000 rounds without any time constraints. At each time step, the agent will perform an attack action. The agent wins if it manages to reach the attack goal within the time-step limit. Two metrics are considered to compare the performance of the agents: winning steps as the number of simulation steps taken to reach their goal, and the cumulative rewards over simulation steps across training epochs. The key results are shown in Fig. 3. We can see that our method significantly outperforms the comparing agents in the learning speed and convergence performance, reaching a performance improvement of nearly 40% across the best baselines. The performance of the MARL-based attacker failed to converge in our tested scenario, since the CTF network environment has strong randomness, which brings great challenges to the agent's exploration ability. In contrast, entropy loss is added in RLAPT to guarantee enough exploration. For clarity, the learning curve of MARL-based attacker is omitted in Fig. 3.

We also conducted experiments on the CTF scenario to understand the effect of the predefined knowledge scale in our method. We adjusted the maximum tolerated node number $N_{\max}$ and the size of the predefined vulnerability libraries of our agent, and tested our method under each different setting. Our agent is trained with the assumption of a larger network system, and then tested with a smaller amount of network nodes. Fig. 4 shows that our method could achieve superior

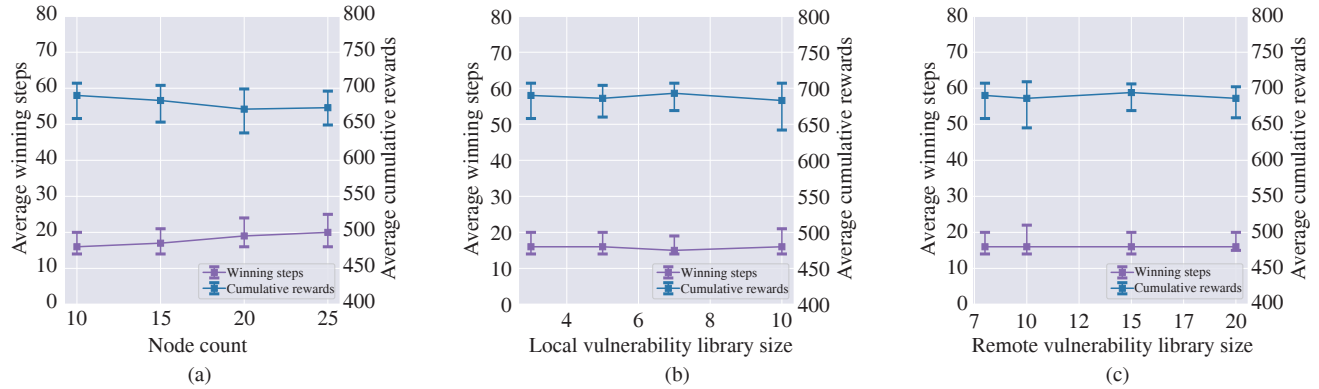


Fig. 4 Comparison between the convergence performance of the RLAPT agents with different knowledge scales. Fig. 4(a) shows the maximum tolerated node number, Fig. 4(b) shows the size of local vulnerability library, and Fig. 4(c) shows the size of remote vulnerability library: (a) Effect of node scale; (b) Effect of local vulnerability scale; (c) Effect of remote vulnerability scale

convergence performance regardless of the predefined knowledge scale.

VI. Conclusion

This paper presents RLAPT, a novel DRL-based method for automated penetration testing that is capable of mastering fine-grained control of the automated penetration testing process, and can be extended more generally to realistic PT scenarios. The test results on simulated scenarios from CyberBattleSim show the superior performance of RLAPT. Future research directions will target the method's robustness and automated penetration testing method that accounts for the defender's existence.

References

[1] RAILKAR D, JOSHI S. A comprehensive literature review of artificial intelligent practices in the field of penetration testing[J]. Intelligent Systems and Applications, 2023: 75-85.

[2] LIPPMANN R, INGOLS K. An annotated review of past papers on attack graphs[R]. [S.l.:s.n.], 2005.

[3] AMMANN P, WIJESEKERA D, KAUSHIK S. Scalable graph-based network vulnerability analysis[C]//Proceedings of the 9th ACM Conference on Computer and Communications Security. New York: ACM Press, 2002: 217-224.

[4] PHILLIPS C, SWILER L P. A graph-based system for network-vulnerability analysis[C]//Proceedings of the 1998 Workshop on New Security Paradigms. New York: ACM Press, 1998: 71-79.

[5] SCHWARTZ J, KURNIAWATI H. Autonomous penetration testing using reinforcement learning[J]. arXiv:1905.05965, 2019.

[6] HU Z, BEURAN R, TAN Y. Automated penetration testing using deep reinforcement learning[C]//Proceedings of 2020 IEEE European Symposium on Security and Privacy Workshops. Piscataway: IEEE Press, 2020: 2-10.

[7] TRAN K, AKELLA A, STANDEN M, et al. Deep hierarchical reinforcement agents for automated penetration testing[J]. arXiv:2109.06449, 2021.

[8] YANG Y, LIU X. Behaviour-diverse automatic penetration testing: a curiosity-driven multi-objective deep reinforcement learning approach[J]. arXiv:2202.10630, 2022.

[9] WANG X, WANG S, LIANG X, et al. Deep reinforcement learning: a survey[J]. IEEE Transactions on Neural Networks and Learning Systems, 2022: 1-15.

[10] LASKIN M, LEE K, STOOKE A, et al. Reinforcement learning with augmented data[J]. Advances in Neural Information Processing Systems, 2020, 33: 19884-19895.

[11] VINYALS O, BABUSCHKIN I, CZARNECKI W, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning[J]. Nature, 2019, 575(7782): 350-354.

[12] STANDEN M, LUCAS M, BOWMAN D, et al. CybORG: a gym for the development of autonomous cyber agents[J]. arXiv:2108.09118, 2021.

[13] SEIFERT C, BETSER M, BLUM W, et al. CyberBattleSim[R]. San Francisco: GitHub, 2021.

[14] LI L, FAYAD R, TAYLOR A. CyGIL: a cyber gym for training autonomous agents over emulated network systems[J]. arXiv:2109.03331, 2021.

[15] FOLEY M, HICKS C, HIGHNAM K, et al. Autonomous network defence using reinforcement learning[C]//Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security. New York: ACM Press, 2022: 1252-1254.

[16] PIPLAI A, ANORUO M, FASAYE K, et al. Knowledge guided two-player reinforcement learning for cyber attacks and defenses[C]//Proceedings of the International Conference on Machine Learning and Applications. Piscataway: IEEE Press, 2022:1342-1349.

[17] WALTER E, FERGUSON W, RIDLEY A. Incorporating deception into CyberBattleSim for autonomous defense[J]. arXiv:2108.13980, 2021.

[18] APPLEBAUM A, DENNLER C, DWYER P, et al. Bridging automated to autonomous cyber defense: foundational analysis of tabular Q-learning[C]//Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security. New York: ACM Press, 2022: 149-159.

[19] SILVER D, SCHRITTWIESER J, SIMONYAN K, et al. Mastering the game of go without human knowledge[J]. Nature, 2017, 550(7676): 354-359.

[20] HAUSKNECHT M, STONE P. Deep recurrent Q-learning for partially observable MDPs[J]. arXiv:1507.06527.

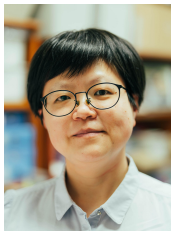
[21] KAELBLING L, LITTMAN M, CASSANDRA A. Planning and acting in partially observable stochastic domains[J]. Artificial Intelligence, 1998, 101(1-2): 99-134.

[22] MNIH V, BADIA A, MIRZA M, et al. Asynchronous methods for deep reinforcement learning[C]//Proceedings of the International Conference on Machine Learning. New York: PMLR, 2016: 1928-1937.

[23] SCHULMAN J, WOLSKI F, DHARIWAL P, et al. Proximal policy optimization algorithms[J]. arXiv:1707.06347, 2017.
[24] TAVAKOLI A, PARDO F, KORMUSHEV P. Action branching architectures for deep reinforcement learning[C]//Proceedings of the AAAI Conference on Artificial Intelligence. Cambridge MA: AAAI, 2018:4131-4138.
[25] YE D, LIU Z, SUN M, et al. Mastering complex control in moba games with deep reinforcement learning[C]//Proceedings of the AAAI Conference on Artificial Intelligence. Cambridge MA: AAAI, 2020: 6672-6679.
[26] BERNER C, BROCKMAN G, CHAN B, et al. Dota 2 with large scale deep reinforcement learning[J]. arXiv:1912.06680, 2019.
[27] SCHULMAN J, MORITZ P, LEVINE S, et al. High-dimensional continuous control using generalized advantage estimation[J]. arXiv:1506.02438, 2015.
[28] SUTTON R, BARTO A. Reinforcement learning: an introduction[M]. Cambridge MA: MIT Press, 2018.

About the Authors

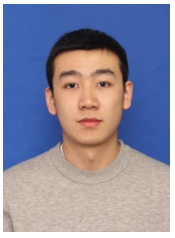
Xiaotong Guo is currently pursuing the M.S. degree with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests include network security, VANET caching, and next-generation mobile communication networks.



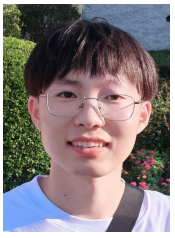
Jing Ren (Member, IEEE) received the B.E. and the Ph.D. degrees in Communication Engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2007 and 2015, respectively. Currently, she is an Assistant Researcher at UESTC. Her research interests include network architecture, protocol design, network modeling, optimization, and network security.

Jiangong Zheng (Student Member, IEEE) is currently pursuing the M.S. degree with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests include WSN routing, congestion control, and next-generation mobile communication networks using reinforcement learning and artificial intelligence technologies.

Jianxin Liao is currently pursuing the M.S. degree with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests include network security, VANET caching, and next-generation mobile communication networks.



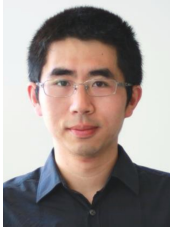
Chao Sun is currently pursuing the M.S. degree with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests include network security, VANET caching, and next-generation mobile communication networks.



Hongxi Zhu is currently pursuing the M.S. degree with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests include network security and next-generation mobile communication networks.

Tongyu Song [corresponding author] received the B.E. and the Ph.D. degrees in Communication Engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2014 and 2022, respectively. Currently, he is an Assistant Researcher at Research Institute of Intelligent Networks, Zhejiang Lab, Hangzhou, China. His research interests include network architecture, artificial intelligence, software-defined networking, and network security.

Sheng Wang received the B.S. degree in Electronic Engineering, and the M.S. and the Ph.D. degrees in Communication Engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 1992, 1995, and 2000, respectively. He is a Professor with the UESTC. His research interests include planning and optimization of wire and wireless networks, next-generation of Internet, and next-generation optical networks.



Wei Wang (Member, IEEE) received the B.S. degree from Central South University, Changsha, China, in 2010, the M.S. degree from Southeast University, Nanjing, China, in 2013, and the Ph.D. degree from the University of New South Wales, Sydney, Australia, in 2017. From 2018 to 2021, he was a Postdoctoral Research Fellow with the School of Electrical Engineering and Telecommunications, the University of New South Wales, Sydney, Australia. He is currently a Senior Research Scientist (equivalent to Associate Professor) with Peng Cheng Laboratory, Shenzhen, China. His current research interests include millimeter-wave communications, wireless localization, machine learning for wireless communications, and cyber security.