Proceedings of the 2nd International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/82/20241051

# Optimizing intelligent penetration path planning using reinforcement learning: A focus on valid action masking and sample enhancement

**Lufan Huang**

Meiji University, Tokyo, Japan

huanglufan9@gmail.com

**Abstract.** As penetration testing is a crucial technique for the rapid evolution of cybersecurity to identify system vulnerabilities, traditional penetration testing methods face the difficulty of large amounts of manual labor and professional knowledge, which makes them difficult to scale. This paper proposes a highly automated and efficient penetration testing approach based on the MASK-SALT-DQN algorithm. MASK-SALT-DQN is a reinforcement learning model that effectively optimizes penetration path planning by valid action masking and sample enhancement. Through valid action masking, the MASK-SALT-DQN model filters out redundant and invalid actions in the solution space, reducing the complexity of the action space and improving the convergence speed and efficiency. In addition, the sample enhancement method increases the frequency of critical exploitation actions and positive rewards in the sparse-reward environment, speeding up the learning process of the agent. Experiments have been conducted on the NASim attack simulation network to demonstrate the advantages of the proposed model over the baseline method. The experimental results show that our model has a significant performance advantage over baseline methods in large networks, and it can be applied to the field of scalable and efficient automated penetration testing.

**Keywords:** Penetration testing, Reinforcement learning, Valid action masking, Sample enhancement, Solution space transformation.

## 1. Introduction

Weaknesses in networked systems can be exposed by simulating actual attacks through penetration testing. In modern cyber-security practice, many enterprises routinely perform penetration testing on their networks. Traditional penetration testing processes are manual and highly dependent on the experience of human experts, which makes them time-consuming and expensive. As a network grows in size and complexity, it becomes increasingly difficult to exhaustively discover all possible vulnerabilities using a manual penetration test. This is why more and more enterprises have begun to explore the use of automated penetration testing using artificial intelligence (AI) and machine learning (ML) to improve both coverage and efficiency. Reinforcement learning (RL) is a sub-field of AI closely related to penetration testing. In particular, the task of finding an optimal penetration path can be modeled as an RL problem. The agent learns to penetrate a complex environment, where the agent's actions are represented by vulnerability scanning, and the reward is determined by the exploitability of the weaknesses. Despite its potential, RL-based penetration testing faces many challenges. The first

DOI: 10.54254/2755-2721/82/20241051

challenge is that the solution space is often filled with a large number of invalid or redundant actions. For example, when scanning a network for vulnerabilities, the agent will repeatedly encounter the same host and thus re-scan it multiple times. Such redundant actions massively increase the complexity of the solution space, making it much harder for the algorithm to identify the optimal path. The second challenge is that the reward is usually sparse in penetration testing. The positive feedback happens infrequently: it takes the agent multiple steps to successfully exploit a vulnerability and penetrate the host. Therefore, the agent will go through multiple iterations without receiving enough rewards for updating the strategy. In this paper, we propose the MASK-SALT-DQN algorithm, which incorporates two innovative ideas: valid action masking and sample enhancement. Valid action masking reduces the complexity of the solution space by filtering out invalid actions. Sample enhancement ensures that sufficient samples of critical actions, such as exploitation of vulnerabilities, are available in the training process [1]. We evaluated the MASK-SALT-DQN algorithm on the NASim network attack simulator and show that it significantly outperforms the baseline models in both convergence speed and accuracy, particularly in large-scale networks.

## 2. Intelligent Penetration Path Planning

### *2.1. Problem Analysis*

The final solution, following the instructions of the planner, must minimise the cost of the penetration goal. The agent starts with an initial state and samples the possible actions. It interacts with the environment and adjusts the strategy accordingly. One of the major flaws is a high redundancy of penetration paths, which makes the solution space more difficult to explore. The reason is that, while planning the attack, the agent can take actions that are ineffective and do not contribute to the target state. It is, for example, the case of trying to exploit vulnerabilities that are not present on a target host, or it could be the case to scan a host that has already been fully explored. The solution space becomes bloated with redundant paths, making it much harder for the algorithm to find the optimal one. Another issue is that success in exploiting any given vulnerability is uncertain. Depending on the type of vulnerability and the defences in place on the target system, some exploitation actions will have a higher success rate than others. Suppose that for each vulnerability, the agent knows the maximum number of times it can attempt to exploit it before it gives up, but the agent doesn't know in advance how long it will take to succeed. This uncertainty kicks in when the agent starts planning sequences of actions [2]. When a vulnerability has a low success rate, the agent must try to exploit it many times before it succeeds in the penetration; the agent can't know in advance how many times it will have to try. This is uncharted territory for planning algorithms. As for the positive rewards that signal successful penetrations, they are few and far between. The agent might have to go through hundreds of iterations of trial and error without receiving a single piece of positive reinforcement.

### *2.2. Solution Space Transformation*

To deal with the issue of dead-ends, we adopt the concept of solution space transformation – a technique that parallels pruning producing a significantly less complex solution space comparable to the state space pruning discussed in the previous section. The original solution space is the set of all possible penetration paths. In this transformation, the original solution space, which contains all possible paths, are all valid steps that lead to meaningful state transitions. The solution space contains many actions that do not contribute to the path to the target state, such as scanning a subnet that has already been explored or carrying out an exploit that does not exist. By removing these invalid actions, the solution space is compressed, leading to ease of finding the optimal path. The transformation is an action that maps a given solution space (the full set of actions) to another solution space (the target space), in which only the valid actions remain. In other words, the action of an agent cannot have a negative influence on reaching the goal [3]. Because the solution space transformation can be measured by a compression ratio, the extent to which this transformation works can be quantified. In the example of the network with multiple subnets and hosts, the number of invalid actions may initially be 90-plus per cent of the action

DOI: 10.54254/2755-2721/82/20241051

space. That computes to a large number of paths that an agent must compute and discard. The solution space transformation can reduce that number exponentially.

### *2.3. Sample Enhancement*

The agent's capacity to take optimal decisions in reinforcement learning is reliant upon the availability of relevant samples in training. In penetration testing, the distribution of relevant samples – especially samples that have led to vulnerability exploitation and positive reward actions – is often sparse. The agent will not see high-impact exploitation opportunities often enough to learn how to evaluate them. For example, an agent may need to perform the same exploitation action a few times in order to learn how effective it can be, but if exploration is random then this action may only be sampled a few times, and be poorly evaluated. To avoid that, we have added a sample enhancement technique to increase the number of samples related to exploitation actions. When exploitation is actually possible (the potential target is vulnerable), the agent repeats exploitation action multiple times to generate more samples. The replay buffer is updated by these additional samples so that the agent has enough data to evaluate the effectiveness of the exploitation. Another enhancement is to use positive reward sample [4]. Because the positive reward information is rare, after the agent actually exploits the vulnerability for one time and gets positive reward, it repeats the exploitation action several times to generate more positive samples. Compared with negative reward only, these positive samples can help the agent learn faster by quickly recognising the actions that lead to effective penetration.

## 3. Experimental Results

### *3.1. Experiment Setup*

We ran a number of experiments on the NASim network attack simulator – an environment, used by researchers, that simulates real-life network environments, making them compatible with reinforcement learning algorithms. Typically, an attacker will traverse the network by scanning services, exploiting vulnerabilities and escalating privileges. The simulator is configurable, allowing an attacker to set the network topology, number of subnets and hosts, and which services are offered on each host, as well as which vulnerabilities exist and what they offer to the attacker. The agent traverses the network topology, having to discover the network configuration by taking actions through the environment [5]. We set up a standard network topology consisting of five subnets – three demilitarised zones (DMZ) and two internal networks – with 16 hosts. The 16 hosts are categorised by their value – assignment to DMZ represents lower value compared to assignment to the internal network. The agent has to penetrate these networks by probing for vulnerabilities and exploiting them to gain root access to the sensitive hosts. The agent has three actions – scanning the services, exploiting a vulnerability and escalating privileges (which he does only if he gains root access to the host on which he exploits a vulnerability). The success rates for each of these actions are dependent on the operating system of the host and the type of vulnerability being targeted. For example, vulnerability exploitation and privilege escalation have a higher success rate on Linux hosts compared with Windows hosts. We also set up larger network topologies (more hosts and subnets) to test the scalability of the algorithm: networks of up to 150 hosts and 30 subnets [6].

The environment is modeled as a Markov Decision Process (MDP), where the state space $S$ consists of all possible configurations of the network. The agent's goal is to transition through the state space to a terminal state $s_T$,where it gains root access to all sensitive hosts. At each step $t$, the agent observes the current state $s_t \in S$ and selects an action $a_t \in A(s_t)$,where $A(s_t)$ represents the set of available actions in state $s_t$.The action $a_t$ is chosen based on the policy $\pi(s_t)$, which is derived using a reinforcement learning algorithm such as $\mathrm{MASK} - \mathrm{SALT} - \mathrm{DQN}$

The agent's actions include:

Service Scanning: Given a host $h_i \in H$, the agent can scan the services $S_i$ running on the host, which updates its knowledge about the host's configuration. The scan success probability $p_{scan}(h_i)$ depends on the network setup and is modeled as a Bernoulli trial.

DOI: 10.54254/2755-2721/82/20241051

Vulnerability Exploitation: For each host $h_i$, the agent attempts to exploit known vulnerabilities $V_i$, where each vulnerability $v_j \in V_i$ has a probability of success $p_{exploit}(v_j)$ based on the host's operating system and security configurations. The action outcome transitions the agent from state $s_t$ to a new state $s_t + 1$, with the reward $R(s_t, a_t)$ defined as a function of the host's sensitivity and the success of the action

Privilege Escalation: Once a vulnerability has been exploited, the agent can escalate its privileges $P_i$ on host $h_i$. The probability of success for privilege escalation $p_{privilege}(h_i)$ depends on the host's internal configurations and the available escalation vectors.

We designed a standard network topology consisting of five subnets and sixteen hosts divided into demilitarized zones (DMZ) and internal networks. The set of subnets $\mathcal{N} - \{n_1, n_2, \ldots, n_5\}$ includes sensitive hosts $H_{sensitive} \subset H$,with higher rewards assigned fon gaining root access on these sensitive hosts. The agent must penetrate these networks by identifying vulnerabilities and exploiting them to gain root access to sensitive hosts. The reward function $R(s_t, a_t)$ is defined as the sum of rewards from all exploited hosts, weighted by their sensitivity

$$R(s_t, a_t) = \sum_{h_i \in H_{senative}} \text{reward}(h_i) \cdot p_{exploit}(v_j) \cdot p_{privilege}(h_i)$$

The success rates of these actions $p_{scan}, p_{exploit}, p_{privilege}$ vary depending on the hosts operating system and the targeted vulnerability. To evaluate the scalability of the algorithm we also created larger network topologies with up to 150 hosts $|H| - 150$ and 30 subnets $|\mathcal{N}| - 30$. The complexity of the state space $|S|$ and action space $|A(s)|$ grows exponentially with the size of the network, making efficient exploration cnucial for optimal performance [7].

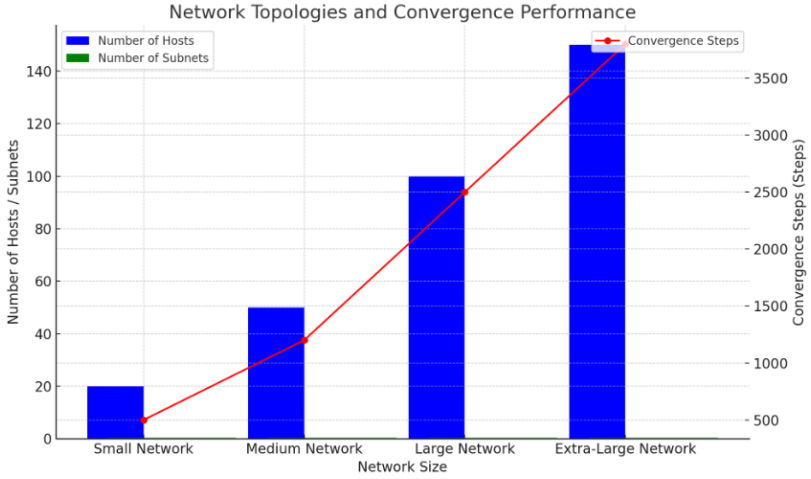


**Figure 1.** Network Topologies and Convergence Performance

## 4. Discussion

### *4.1. Valid Action Masking*

A key component of the MASK-SALT-DQN algorithm is the use of valid action masking, which eliminates invalid actions at each decision point. The agents will consider only actions taking the Q-values of invalid actions to -inf Fully exploiting the network as quickly as possible is paramount, and the use of valid action masking helps achieve this. In our environment we considered invalid actions as non-sensible actions like exploiting vulnerabilities that do not exist in the current host and scanning subnets that have already been fully explored. By performing such actions, agents would deplete their time without delivering any progress [8]. Table 1 shows an example of how the algorithm eliminates the action of exploiting vulnerability C in host 129 since it is not a valid action. As can be seen, the agent

DOI: 10.54254/2755-2721/82/20241051

only considers valid actions, by setting the Q-values of invalid actions to -inf. This encourages the agent to explore the network quickly, as it does not waste time exploring useless paths. In conclusion, valid action masking ensures that the agents focus on the actions relevant to the current states, thus reducing the number of steps needed to reach the optimal penetration path [9].

**Table 1.** Valid Action Masking Process in the MASK-SALT-DQN Algorithm

| State (Host) | Available Actions | Valid Actions (After Masking) | Action Value (Q-Value) | Masked Q-Value |
|---|---|---|---|---|
| Host A | Scan Service, Exploit Vuln X, Exploit Vuln Y | Scan Service, Exploit Vuln X | 0.85, 0.75, -inf | 0.85, 0.75, -inf |
| Host B | Scan Service, Exploit Vuln Z, Escalate Priv. | Exploit Vuln Z, Escalate Priv. | -inf, 0.90, 0.65 | -inf, 0.90, 0.65 |
| Host C | Scan Service, Exploit Vuln W, Exploit Vuln V | Exploit Vuln W | -inf, 0.80, -inf | -inf, 0.80, -inf |
| Host D | Scan Service, Escalate Priv., Exploit Vuln T | Scan Service, Exploit Vuln T | 0.70, -inf, 0.60 | 0.70, -inf, 0.60 |
| Host E | Exploit Vuln Q, Escalate Priv., Scan Service | Escalate Priv., Scan Service | -inf, 0.78, 0.68 | -inf, 0.78, 0.68 |

### *4.2. Compression Ratio Impact*

The compression ratio gives a quantitative measure of the quality of the solution space transformation: by removing redundant actions, the action space becomes smaller and it becomes easier for the algorithm to find the optimal path. The experiments showed that the impact of the solution space transformation becomes more evident in bigger networks – where the action space can become quite overwhelming. For example, in a network with 100 hosts and 20 subnets, the compression ratio equals to 57.2 [10]. This means that the size of the mapped solution space is about 57 times smaller than the original one – an exceptional improvement also in terms of complexity. In Table 2 below, the effect of the compression ratio on the size of the action space and on convergence is highlighted: in bigger networks, the impact is even more evident as there is more room for the transformation to truly reduce redundancy in the paths, and therefore the size of the solution space becomes considerably smaller.

**Table 2.** Impact of Compression Ratio on Solution Space Efficiency

| Network Size | Number of Hosts | Number of Subnets | Original Action Space | Transformed Action Space | Compression Ratio | Convergence Speed (Steps) |
|---|---|---|---|---|---|---|
| Small Network | 20 | 5 | 10,000 | 2,000 | 5.0 | 500 |
| Medium Network | 50 | 10 | 50,000 | 8,333 | 6.0 | 1,200 |
| Large Network | 100 | 20 | 250,000 | 4,375 | 57.2 | 2,500 |
| Extra-Large Network | 150 | 30 | 1,000,000 | 14,285 | 70.0 | 3,800 |

## 5. Conclusion

This paper describes a recently improved version of the MASK-SALT-DQN algorithm that overcomes the redundancy in the path space and sparsity of rewards in automated penetration testing, by applying valid action masking and sample enhancement. Valid action masking filters out unnecessary actions, reducing the complexity of the search space and facilitating the absorption of useful information as the agent focuses only on relevant actions. Sample enhancement ensures that critical actions and positive rewards are adequately sampled, accelerating the learning process in the sparse-reward environment. In

DOI: 10.54254/2755-2721/82/20241051

our experiments on several network topologies using the NASim network attack simulator, our algorithm outperformed the baseline methods significantly. When MASK-SALT-DQN was applied to a network-blocked network with 800 hosts, the agent successfully found penetration paths using only one-third of the trials compared with another climbing algorithm. Our results demonstrate the potential of RL models such as MASK-SALT-DQN for automatically testing the security and reliability of vulnerabilities in large networks, making it more efficient and scalable.

**References**

[1] Altulaihan, Esra Abdullatif, Abrar Alismail, and Mounir Frikha. "A survey on web application penetration testing." Electronics 12.5 (2023): 1229.

[2] Greco, Claudia, et al. "AI-enabled IoT penetration testing: state-of-the-art and research challenges." Enterprise Information Systems 17.9 (2023): 2130014.

[3] Heiding, Fredrik, et al. "Penetration testing of connected households." Computers & Security 126 (2023): 103067.

[4] Ghanem, Mohamed C., Thomas M. Chen, and Erivelton G. Nepomuceno. "Hierarchical reinforcement learning for efficient and effective automated penetration testing of large networks." Journal of Intelligent Information Systems 60.2 (2023): 281-303.

[5] Almazrouei, Omar, et al. "Penetration Testing for IoT Security: The Case Study of a Wireless IP Security CAM." 2023 IEEE 2nd International Conference on AI in Cybersecurity (ICAIC). IEEE, 2023.

[6] Ojha, Avinash, and Praveen Aggarwal. "Durability performance of low calcium Flyash-Based geopolymer concrete." Structures. Vol. 54. Elsevier, 2023.

[7] Shakya, Ashish Kumar, Gopinatha Pillai, and Sohom Chakrabarty. "Reinforcement learning algorithms: A brief survey." Expert Systems with Applications 231 (2023): 120495.

[8] Moerland, Thomas M., et al. "Model-based reinforcement learning: A survey." Foundations and Trends® in Machine Learning 16.1 (2023): 1-118.

[9] Casper, Stephen, et al. "Open problems and fundamental limitations of reinforcement learning from human feedback." arXiv preprint arXiv:2307.15217 (2023).

[10] Ball, Philip J., et al. "Efficient online reinforcement learning with offline data." International Conference on Machine Learning. PMLR, 2023.