# 基于启发式奖赏塑形方法的智能化攻击路径发现

曾庆伟, 张国敏, 邢长友, 宋丽华

陆军工程大学 指挥控制工程学院 南京 中国 210007

**摘要** 渗透测试作为一种评估网络系统安全性能的重要手段, 是以攻击者的角度模拟真实的网络攻击, 找出网络系统中的脆弱点。而自动化渗透测试则是利用各种智能化方法实现渗透测试过程的自动化, 从而大幅降低渗透测试的成本。攻击路径发现作为自动化渗透测试中的关键技术, 如何快速有效地在网络系统中实现智能化攻击路径发现, 一直受到学术界的广泛关注。现有的自动化渗透测试方法主要基于强化学习框架实现智能化攻击路径发现, 但还存在奖赏稀疏、学习效率低等问题, 导致算法收敛速度慢, 攻击路径发现难以满足渗透测试的高时效性需求。为此, 提出一种基于势能的启发式奖赏塑形函数的分层强化学习算法(HRL-HRSF), 该算法首先利用渗透测试的特性, 根据网络攻击的先验知识提出了一种基于深度横向渗透的启发式方法, 并利用该启发式方法设计出基于势能的启发式奖赏塑形函数, 以此为智能体前期探索提供正向反馈, 有效缓解了奖赏稀疏的问题; 然后将该塑形函数与分层强化学习算法相结合, 不仅能够有效减少环境状态空间与动作空间大小, 还能大幅度提高智能体在攻击路径发现过程中的奖赏反馈, 加快智能体的学习效率。实验结果表明, HRL-HRSF 相较于没有奖赏塑形的分层强化学习算法、DQN 及其改进算法更加快速有效, 并且随着网络规模和主机漏洞数目的增大, HRL-HRSF 均能保持更好地学习效率, 拥有良好的鲁棒性和泛化性。

**关键词** 自动化渗透测试; 奖赏塑形; 分层强化学习; 攻击路径发现; DQN 算法

**中图法分类号** TP393 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2024.05.04

# Intelligent Attack Path Discovery Based on Heuristic Reward Shaping Method

ZENG Qingwei, ZHANG Guomin, XING Changyou, SONG Lihua

College of Command and Control Engineering, Army Engineering University, Nanjing 210007, China

**Abstract** As an important means to evaluate the security performance of network systems, penetration testing is to simulate real network attacks from the perspective of attackers and find out the vulnerable points in network systems. The automatic penetration test uses various intelligent methods to realize the automation of the penetration test process, thus greatly reducing the cost of penetration test. Attack path discovery is a key technology in automated penetration testing. how to quickly and effectively implement intelligent attack path discovery in network systems has been widely concerned by the academic community. The existing automated penetration testing methods are mainly based on the reinforcement learning framework to achieve intelligent attack path discovery. However, there are still problems such as sparse rewards and low learning efficiency, which lead to slow convergence of the algorithm. Attack path discovery cannot meet the high timeliness requirements of penetration testing. Therefore, a layered reinforcement learning algorithm (HRL-HRSF) based on potential energy heuristic reward shaping function is proposed. This algorithm first uses the characteristics of penetration testing to propose a heuristic method based on depth horizontal penetration according to the prior knowledge of network attacks, and uses this heuristic method to design a potential energy based heuristic reward shaping function to provide positive feedback for early exploration of agents, it effectively alleviates the problem of sparse rewards. Then combining the shaping function with hierarchical reinforcement learning algorithm can not only effectively reduce the size of environment state space and action space, but also greatly improve the reward feedback of agents in the process of attack path discovery, and accelerate the learning efficiency of agents. The experimental results show that HRL-HRSF is faster and more effective than layered reinforcement learning algorithm without reward shaping, DQN and its improved algorithm. With the increase of network size and host vulnerabilities, HRL-HRSF can maintain better learning efficiency, has good robustness and generalization.

**Key words** automated penetration testing; reward shaping; hierarchical reinforcement learning; attack path discovery; DQN algorithm

**通讯作者:** 张国敏, 博士, 副教授, Email: zhang_gmwn@163.com。

本课题得到国家自然科学基金(No. 62172432)资助。

收稿日期: 2022-07-31; 修改日期: 2022-11-26; 定稿日期: 2024-01-17

## 1 引言

渗透测试(Penetration Testing, PT)是一种通过模拟黑客攻击, 在不影响目标系统网络的前提下, 利用系统漏洞, 获得系统控制权的网络安全测试方法[1-2]。通过上述方法, 目标系统即可根据报告中的漏洞, 采取相应的防护措施, 防范于未然, 使得系统网络更加健壮, 不易被黑客攻击。但是现代企业或者组织的计算机网络系统庞大且复杂, 其中运行的各种软件还在不断发生更新、修改、删除、迁移等变化, 在渗透测试过程中会存在大量重复流程, 单纯依赖安全专家和安全分析人员来实施安全测试不切实际, 需要耗费大量的时间和精力, 加剧了渗透测试的代价。因此, 利用智能的自动化渗透测试技术来模拟具有攻击策略的真实攻击者, 找出目标系统网络中的关键攻击路径, 发现网络中的薄弱点, 从而可以大幅减少渗透测试成本, 提高渗透测试效率。

强化学习(Reinforcement Learning, RL)[3]是一种学习范式, 通过试错来获取经验知识, 并根据经验学习在环境中采取行动的最佳策略, RL 自主学习和在线学习的特点使其在各种领域中都取得了很多重要成果, 同时基于强化学习的攻击路径发现相关研究也成为了相关领域中研究的热点。RL 的本质决定了其面临的维度“灾难”问题无法从根本上消除, 但是可以从方法上加以克服。其中分层强化学习(Hierarchical Reinforcement Learning, HRL)就是一种重要方法, HRL 本质上就是把一个复杂的任务分解为不同抽象层次上的子任务, 子任务相对于整体目标动作空间大幅减少, 可以更快地进行求解, 从而加快整体目标的求解速度[4]。尽管 HRL 算法有利于从更高级的领域知识来指导智能体, 但是在训练初期同样会面临盲目搜索的问题, 从而导致很多无效的操作, 延缓了算法训练速度, 而对于攻击路径发现技术来说, 算法的运行速度与渗透测试的结果息息相关。因为在实际渗透测试环境中, 网络状态更为复杂, 网络信息的获取方式更为多样, 各种自动化攻击工具之间的联动更为普遍, 这在人工智能的训练中也就需要更多的训练参数, 更长的训练时间, 而在网络安全领域中, 网络环境瞬息万变, 渗透测试更是注重时效性, 攻击路径更是需要对实时的网络状态的准确反映, 长时间的路径搜索会对攻击路径的准确性产生很大影响。因此, 结合人工知识经验的智能化攻击路径发现技术来模拟具有攻击策略的真实攻击者, 将人工先验知识转化为在某些状态下应该执行哪些动作的启发式方法, 通过人工经验指导学习, 减少智能体在训练初期的一些无用操作, 指导后续的学习过程, 提高人工智能的训练效率, 加速智能策略的生成, 从而可以执行更有效的渗透攻击。

总之, 目前的攻击路径发现研究还存在以下挑战: 1) 人工搜索攻击路径需要耗费大量的时间成本; 2) 基于 RL 的攻击路径发现方法面临着维度“爆炸”问题; 3) 基于 HRL 的攻击路径发现的盲目探索问题限制了算法的训练效率。

为了解决以上挑战, 我们提出了一种基于启发式奖赏塑形函数的分层强化学习算法, 实现了在未知渗透测试环境下攻击路径的快速发现。本文的主要贡献如下:

1) 为了解决分层强化学习算法前期学习效率低的问题, 本文基于渗透测试的特性, 提出了一种基于主机优先横向渗透的启发式设计, 以此为先验知识指导智能体优先渗透不同子网的跨网段主机, 从而实现更快地渗透到目标主机。

2) 本文利用基于先验知识的启发式信息构造奖赏塑形函数, 将基于势能的奖赏塑形方法引入到分层强化学习算法中, 并通过对人工经验推荐的动作给予额外奖赏, 以鼓励智能体优先选择这些动作, 实现在不改变原始最优策略的情况下指导智能体的探索过程。

3) 基于开源的网络攻击模拟器 NASim 构建的网络场景来用于智能体训练, 通过对比试验, 验证了我们的方法在性能上的优势, 并且通过改变网络规模, 验证了该方法的可扩展性, 最后利用预训练的智能体在网络环境改变的情况下验证了算法的泛化能力。

## 2 相关工作

近年来, 强化学习已经取得了显著的成功, 在一些游戏中, 如 Alpha Go[5]、Alpha Star[6]等, 基于强化学习的代理已经超越了人类玩家。与许多游戏规则类似, 渗透测试也是一个基于环境状态的动态决策过程。基于强化学习的自动化渗透测试方法大致可分为基于传统强化学习的渗透测试和基于深度强化学习的渗透测试。基于传统强化学习的渗透测试将攻击过程建模为部分可观察马尔可夫决策过程(Partially Observable Markov Decision Process, POMDP)[7], 这种方法可以更好地模拟真实渗透测试场景下攻击者对目标环境的不确定性, 但由于 POMDP 的局限性, 导致该方法的计算复杂度随着问题规模呈指数增长, 一般只适用于求解小规模问题。

基于深度强化学习的方法通常将渗透测试过程形式化为马尔科夫决策过程(Markov Decision Process, MDP), 通过设计动作执行的成功概率, 可以在一定程度上模拟现实世界中攻击者对目标网络系统的不确定性, 通过设计奖惩机制, 引导训练智能体根据环境状态选择最佳动作。Zhou 等[8]在强化学习的奖励值设计中引入信息熵, 并在此基础上提出基于网络信息增益的攻击路径发现算法, 利用网络信息来获取奖励并指导智能体进行最优动作选择。但是该方法主要解决的是单主机的攻击路径发现, 无法应用于大规模的网络环境。为了解决该问题, 周仕承等人[9]将深度强化学习与渗透测试问题相结合, 将渗透测试问题建模为 MDP 模型, 提出了一种改进的深度强化学习算法 Noisy-Double-Dueling DQNper, 实现了较大网络规模的攻击路径发现。同时为了解决大型动作空间问题, Zhou 等[10]进一步提出了一种改进的 DQN 算法——NDSPI-DQN 进行攻击路径发现, 该方法通过解耦攻击向量, 有效减少了代理的动作空间。利用 MDP 模型可以很好的模拟渗透测试过程, 但是强化学习面临的探索效率低的问题依然存在, 并且对于渗透测试问题来说, 只有在攻击者成功入侵敏感主机时才会获得正向奖赏, 因此在大规模网络环境中, 奖赏稀疏问题尤为突出。一种解决该问题最为直接且有效的方法就是将人工先验知识引入到智能体学习过程中[11-12]。

面向渗透测试的攻击路径发现问题作为一种专业性非常强的研究领域, 专家经验是非常重要的先验知识, 利用先验知识可以很好的引导算法进行路径搜索, 因此将领域相关的经验知识引入攻击路径发现过程中, 可以起到事半功倍的效果。胡泰然等[13]将漏洞威胁程度, 漏洞成功率以及主机资产作为启发式信息来构建启发式函数, 利用启发信息引导攻击路径发现, 不仅提高了路径发现的效率, 更是提高了攻击路径的可行性。因此针对基于强化学习的智能化攻击路径发现问题, 可以利用渗透测试领域中的专业知识设计启发式信息, 应用于智能体的学习过程中, 从而解决智能化攻击路径发现所面临的奖赏稀疏问题, 以加快强化学习算法的训练效率。

而在强化学习中, 将先验知识引入智能体的学习过程中比较典型的主要有两种方法。

其中一种方法就是逆强化学习(Inverse Reinforcement Learning, IRL), 它根据人类提供的最优策略或轨迹, 推断出未观察到的奖赏函数, 从而继续进行传统的强化学习。Ng 和 Russell[14]提出了从最优行为样本中学习奖赏函数需要满足的约束条件, 推动了大量逆强化学习算法和应用的出现。然而对于渗透测试问题, 攻击者在渗透目标网络之前, 是无法获取目标网络相关的状态信息和样本经验, 因此也就无法根据已有的策略来推断奖赏函数。

另一种有效方法就是奖赏塑形(Reward Shaping), 它通过提供一个易于学习的奖赏函数来引入先验知识, 减少智能体在探索过程中的随机性, 以加快智能体更快地学习到最优策略。Ng 等[15]指出, 当塑形函数是基于势能的塑形函数, 不会影响原始问题的最优策略。Marom 和 Rosman[16]提出了基于贝叶斯方法的奖赏塑形, 称为信念奖赏塑形(BRS), 环境奖赏对状态和动作的估计概率分布的期望值被用作塑造奖赏, 他们证明了通过使用 BRS 增强的 Q-learning 学习到的策略与原始 MDP 上的最优策略一致。随后, 研究出了更多不改变最优策略的势函数形式, 包括动态环境下基于值的势函数[17]和基于状态-动作值的势函数[3]。Gao 和 Toni[18]等还对奖赏塑形在 HRL 中的应用进行了研究, 并证明了基于势能的奖赏塑形对 MAX-Q 算法保持策略不变。

综合以上分析, 现有的基于强化学习的攻击路径发现问题还存在奖赏稀疏、学习效率低的问题, 基于逆强化学习的方法需要已知原始问题的最优策略而不适用于攻击路径发现, 基于奖赏塑形的方法通过利用辅助控制来增强智能体在探索过程中的动作奖赏, 在加快智能体学习的情况下而不影响算法最优策略, 因此本文研究奖赏塑形方法与分层强化学习算法的结合, 通过在攻击路径发现技术中的应用, 解决攻击路径发现在大规模网络中探索效率低的问题, 从而提高现有智能化攻击路径发现算法的效率。

## 3 模型与算法理论

### 3.1 分层强化学习

分层强化学习(Hierarchical Reinforcement Learning, HRL)是为解决强化学习维度灾难问题而提出的, 本质上就是把一个复杂的任务分解为不同抽象层次上的子任务, 子任务相对于整体目标动作空间大幅减少, 可以更快地进行求解, 从而加快整体目标的求解速度[4]。它是以半马尔科夫决策过程(Semi-Markov Decision Process, SMDP)[19]为模型基础, 就是通过将 MDP 中的连续性状态变化过程分解为间歇性状态变化过程, 以解决一些需要多个时间步才能体现其动作价值的任务。如图 1 所示, 在 SMDP 模型中, 根据每个动作之间的时间间隔(整数或实数), 可以将 SMDP 细分为离散时间 SMDP[19]和连续时间 SMDP[20]。

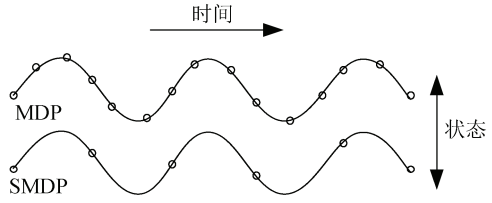


**图 1 MDP 与 SMDP**
**Figure 1 MDP and SMDP**

在 SMDP 中, 设 $\tau$ 是系统在状态 $s$ 执行动作 $a$ 后的随机等待时间, $P(s',\tau|s,a)$ 是从状态 $s$ 执行动作 $a$ 转移到状态 $s'$ 的状态转移函数, 奖赏值为 $R(s,a)$。则状态值函数和状态-动作值函数 Bellman 最优方程为:

$$V^*(s)=\max_{a\in A}[R(s,a)+\sum_{s',\tau}\gamma^{\tau}P(s',\tau|s,a)V^*(s')] \quad (1)$$

$$Q^*(s,a)=R(s,a)+\sum_{s',\tau}\gamma^{\tau}P(s',\tau|s,a)\max_{a'\in A_{s'}}Q^*(s',a') \quad (2)$$

## 3.2 奖赏塑形

大多数基于价值的 RL 算法学习速度较慢的原因之一是它们没有利用先验知识来加快学习。因此, 强化学习算法最初能做的就是随机均匀地探索状态-动作对, 只有在观察到足够的状态转换关系及其相应奖赏之后, 智能体才能开始利用这些经验知识, 并将其动作选择策略改变为选择偏向于其估值偏好的动作。

奖赏塑形是一种将先验知识引入学习过程以缓解这一问题的主流方法。它为智能体学习提供额外的辅助奖励, 这个额外的奖赏可以丰富稀疏的基本奖赏信号, 为智能体提供有用的梯度信息。该辅助塑形函数 $F$ 被添加到环境的奖赏函数 $R$ 中, 以构造为一个新的复合奖赏函数, 新的奖赏函数 $R'$ 可以写成如下形式:

$$R'(s,a,s')=R(s,a,s')+F(s,s') \quad (3)$$

其中 $R'$ 表示改变后的新奖赏函数, $R$ 表示原始奖赏函数, $F$ 表示塑形函数。奖赏塑形函数通常根据领域的先验知识构建而成, 并且在很多复杂环境中都得到了应用[21]。同时为了保证塑形函数的引入不影响原始问题的最优策略, Ng 等人[15]提出了一种基于势能的奖赏塑形函数(Potential-Based Reward Shaping, PBRS), 通过把塑形函数分解为基于状态的势函数之差的形式, 如式 4 所示, 即可保证最优策略保持不变。

$$F(s,a,s')=\gamma\ \Phi(s')-\Phi(s) \quad (4)$$

但是在 PBRS 中, 由于势能值仅基于状态, 因此这些值无法提供有关在某些状态下哪些动作更有希望的指示。为了解决这个问题, Wiewiora[3]等人进一步将 PBRS 应用到基于动作的塑形函数中, 如 $F(s,a,s',a')=\gamma\ \Phi(s',a')-\Phi(s,a)$; 之后 Devlin 和 Kudenko[17]等人还证明了基于时间变化的势函数也是有效的, 比如 $F(s,t,s',t+1)=\gamma\ \Phi(s',t+1)-\Phi(s,t)$。

## 3.3 渗透测试网络模型

基于强化学习的渗透测试建模问题目前尚处于研究的初步阶段, 基于POMDP的建模方法受其计算复杂度的限制只适用于小规模问题, 基于 MDP 的建模方法还存在状态和动作空间爆炸至成百上千维的挑战。而在渗透测试过程中, 攻击路径是目标网络中存在的可以被攻击者用于获取特定资产的漏洞序列, 而攻击路径发现, 是从目标网络中发现所有这些漏洞序列的技术, 不仅包括了单主机的攻击过程, 还包括了主机之间的横向渗透路径。因此本文将主机间横向渗透路径和单主机攻击路径进行解耦, 利用分层的 MDP 模型建模该过程。具体的建模过程如下所示。

### 3.3.1 上层主机间横向渗透过程建模

分层的MDP模型分别用四元组 $\{S,A,R,T\}$ 表示, $S$ 为网络状态空间, $A$ 为智能体的动作空间, $R$ 表示智能体采取动作后获取的奖赏, $T$ 为状态转移函数。对于上层 MDP, 状态空间 $S$ 定义为当前网络中每一台主机是否为受控节点的状态, 用一个一维的布尔向量表示, 对于一个 $M$ 台主机的网络, 第 $i$ 台主机的值 $r_i\in\{0,1\}$, 当 $r_i=0$ 时, 主机未被控制, 当 $r_i=1$ 时, 主机 $i$ 被控制。动作空间 $A$ 定义为选择下一步攻击的目标主机, 对于一个 $M$ 台主机的网络, 其动作空间大小为 $O(M)$。奖赏函数 $R$ 决定了能否找到一条主机之间的最佳渗透路径, 我们将奖赏函数定义为所有已成功渗透的主机价值减去所有动作的成本, 其中主机的价值代表的是不同主机的重要程度, 对于敏感主机而言, 主机价值可以设置为一个较大的正数, 对于不同的动作(如扫描、漏洞利用、本地提权等)定义为不同的代价, 动作的代价是为时间、技能、金钱成本的综合量化值。因此, 如果没有主机被破坏, 奖赏就是执行操作的代价。如式 5 所示, $H$ 为所有被智能体入侵的主机集合, $value(h)$ 为成功渗透的主机价值, $A$ 表示智能体采取的所有动作的集合, $cost(a)$ 为动作 $a$ 的代价。

$$R=\sum_{h\in H}value(h)-\sum_{a\in A}cost(a) \quad (5)$$

上层转移函数 $T$ 决定了网络中主机的受控情况在智能体执行主机选择动作之后如何随着训练步数进行演变的过程。具体来说, 因为上层主机间横向渗

透过程表示的是当前网络中所有主机被入侵的情况, 所以其下一状态的转变与下层单主机的攻击是否成功有关, 当下层针对当前主机的攻击成功, 则上层状态空间中该主机的受控标志位设置为“1”, 否则设置为“0”。例如, 对于 4.1 节中图 4 中的网络, 假设当前智能体已成功入侵主机(1,0)和(2,0), 则此时的状态表示为(1,0,1,0,0,0,0), 若下一步上层 MDP 选择主机(3,2)进行攻击, 且下层 MDP 攻击成功的话, 则此时的状态表示转移为(1,0,1,0,0,0,1)。

### 3.3.2 下层单主机攻击路径发现建模

对于下层 MDP, 状态空间 $S$ 定义为上层选择的主机的所有状态信息和该主机的位置标识信息, 如图 2。状态信息包括主机标识、操作系统标识、服务信息和控制信息, 其中主机标识由子网号和主机号唯一标识, 操作系统标识表示主机的操作系统, 服务信息表示主机运行的可以被漏洞利用的服务, 控制信息包括当前主机是否被攻陷、是否可访问、是否被发现、主机的资产价值、可被访问的权限级别(USER 或 ROOT); 当前主机的位置标识表示该主机位于所有主机中的位置。以上信息通过 one-hot 方式编码, 主机标识的维度等于子网数和各子网最大主机数之和, 操作系统标识维度等于网络中所有操作系统种类数, 服务信息维度等于网络中所有运行的服务种类数, 控制字段维度固定; 位置标识维度等于网络所有主机总数。例如, 4.1 节中图 4 的网络共有 7 台主机, 则主机(1,1)对应的位置标识为(0,1,0,0,0,0,0)。

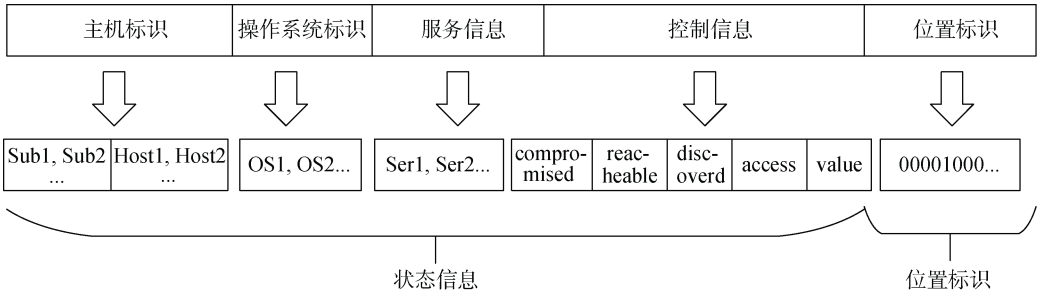


**图 2 状态空间向量化表示**

**Figure 2 State space vectorized representation**

动作空间 $A$ 表示下层策略可以采取的所有攻击动作, 为了模拟真实攻击者的行为, 我们假设代理不能直接获取网络拓扑信息和主机配置信息。因此, 除了漏洞利用和权限提升动作外, 还可以通过扫描来获取主机和目标网络的相关信息。其中扫描操作包括主机扫描和子网信息扫描, 其中, 子网信息扫描用来发现新的子网及其中的主机, 主机扫描是为了获取某一主机的配置信息; 进一步可分为操作系统扫描和服务信息扫描, 通过操作系统扫描, 可对状态空间中操作系统字段进行更新, 通过服务信息扫描, 可确定主机运行了哪些服务, 进而帮助攻击者选择最佳漏洞利用动作进行渗透利用。漏洞利用操作是攻击者针对主机某一存在漏洞的服务, 选择合适的参数和漏洞利用程序进行攻击。许多自动化渗透测试工具已集成了大量成熟的漏洞利用程序, 因此本文对漏洞利用动作进行抽象表示, 忽略了选择参数和漏洞利用程序的过程, 认为针对特定服务执行相应的动作后可以以一定概率攻下主机。

我们选择攻击者在渗透攻击过程中经常使用的进程和服务来代替网络安全漏洞, 同时为了模拟现实世界中攻击的不确定性, 每个操作都有一个成功概率, 其中扫描和权限提升的概率设置为“1”, 漏洞利用的成功概率根据通用漏洞评估系统(Common Vulnerability Scoring System, CVSS)确定。CVSS 中有多类评分细则, 其中攻击复杂度(Attack Complexity, AC)指标用于评价漏洞利用的难易程度, AC 有三个取值分别为高(High)、中(Medium)和低(Low), 其中低复杂度的漏洞表示攻击者可以较为容易的攻击, 漏洞利用所需的条件比较少, 而高复杂度的漏洞成功利用所需的前提条件较多, 攻击者需要对目标进行大量的准备。因此为了刻画漏洞利用的不确定性, 在漏洞利用的成功概率设定上, 本文采用与 Backes 等人[22]相同的方法, 基于 CVSS 中的 AC 指标, 按照指标的“高”“中”“低”值, 分别设置为“0.2”“0.5”“0.8”的概率值。

下层奖赏函数决定了能否找到一条针对单主机的最佳动作序列。因此, 我们将奖赏函数设置为智能体是否成功攻陷目标主机, 如式 6 所示, 当下层网络在状态 $s$ 下采取动作 $a$ 后转移至状态 $s'$, 如果 $s'$ 对应的单主机不是目标状态, 则动作所获得的立即奖赏为 0; 如果 $s'$ 对应的状态是目标状态, 则立即奖赏设置为 1。即当在规定步数内获取主机的 ROOT 权限,

将奖赏值设置为 1, 否则奖赏值设置为 0。

$$r=\begin{cases}1, \textit{ target state}\\ 0, \textit{ other state}\end{cases} \tag{6}$$

下层转移函数 $T$ 决定了网络中某一台主机的状态信息随着攻击动作的选择而转移的过程。对于每一台主机, 其下一个状态的转移取决于攻击操作是否成功, 而这又取决于很多因素。具体来说, 当前攻击的主机是否可达; 攻击动作是扫描还是漏洞利用操作; 对于不同子网间的横向渗透过程中, 受控主机和目标主机之间的目标服务(即可利用的漏洞)流量是否被防火墙限制; 目标主机是否正在运行目标服务; 以及攻击操作的成功概率等。图 3 是一个状态转移过程的示例, 其中智能体成功利用了主机(1,0)上的HTTP服务漏洞, 新状态通过将主机(1,0)的受控标志设置为 *true* 来表示攻击成功。

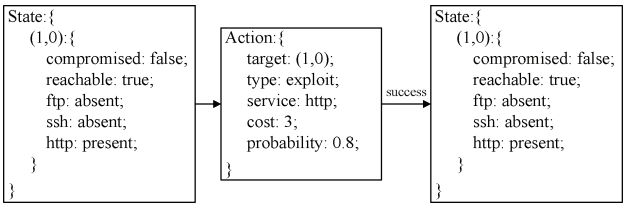


**图 3　单主机状态转移过程示例**

**Figure 3　Example of single host state transition process**

## 3.4　基于分层强化学习的攻击路径发现算法

### 3.4.1　基于 DQN 的上层主机渗透路径发现算法

在基于分层强化学习的攻击路径发现的上层任务中, 主要目标是进行主机之间渗透的攻击规划, 根据当前受控主机的状态, 选择下一步攻击的目标主机, 使得智能体快速且没有重复攻击地实现所有敏感主机的渗透, 完成路径规划任务, 此任务的 MDP 具体见 3.3 节的形式化表示。在该层中使用基于 DQN 的深度强化学习算法, 将经验存储在经验池 $D_1$ 中, 经验池中存储的历史经验为五元组信息 $(s_t, g_t, F_{t\sim t+N}, s_{t+N}, V_t)$, $s_t$ 为当前状态, $g_t$ 为当前时刻输出的目标主机, $F_{t\sim t+N}$ 为接下来 $N$ 个时刻智能体获得的奖赏总和, $s_{t+N}$ 为 $N$ 个时刻后的状态, $V_t$ 为当前时刻是否已经攻陷所有敏感主机。损失函数为:

$$L_1(\theta_{1,i})=E_{(s_t,g_t,F_{t\sim t+N},s_{t+N},V_t)\sim D_1}[(y_{1,i}-Q_1(s,g;\theta_{1,i}))^2] \tag{7}$$

其中 $i$ 表示训练迭代次数, 目标函数 $y_{1,i}=R+\lambda\max_{g'}Q_1(s',g';\theta_{1,i})$。

神经网络参数 $\theta_{1,i}$ 的更新公式为:

$$\nabla_{\theta_{1,i}}L_1(\theta_{1,i})=E_{(s,g,F,s',D)\sim D_1}[(f+\gamma max_{g'}Q_1(s',g';\theta_{1,i-1}) -Q_1(s,g;\theta_{1,i}))\nabla_{\theta_{1,i}}Q_1(s,g;\theta_{1,i})] \tag{8}$$

状态动作值函数的贝尔曼最优方程为:

$$Q_1^*(s,g)=max_{\pi_g}E[\sum_{t'=t}^{t+N}f_{t'}+\gamma max_{g'}Q_1^*(s_{t+N},g')\big|s_t=s, g_t=g,\pi_g] \tag{9}$$

其中 $g'$ 为状态 $s_{t+N}$ 下的目标主机, $\pi_g=P(g|s)$ 是目标选择策略。

### 3.4.2　基于 DQN 的下层单主机攻击动作选择算法

在下层单主机的攻击规划任务中, 主要目标是实现单主机的攻击动作规划, 根据当前主机的状态信息, 选择一系列攻击动作, 完成单主机的渗透。我们将主机的受控状态作为下层的子目标, 同样在该层也使用 DQN 算法, 将经验存储在经验池 $D_2$ 中, 经验池中存储的历史经验为五元组信息 $((s_t,g_t),a_t,r_t,(s_{t+1},g_t),v_t)$, $s_t$ 为当前状态, $g_t$ 为当前需要完成的子目标, $a_t$ 为执行的动作, $r_t$ 为执行动作 $a_t$ 到达下一状态获得的下层奖赏, $s_{t+1}$ 为执行动作 $a_t$ 后的下一个状态, $v_t$ 为是否已经攻陷目标主机。损失函数和神经网络参数 $\theta_{2,j}$ 更新公式为:

$$L_1(\theta_{2,j})=E_{((s_t,g_t),a_t,r_t,(s_{t+1},g_t),v_t)\sim D_2}[(y_{2,j}-Q_2(s,a;\theta_{2,j},g))^2] \tag{10}$$

$$\nabla_{\theta_{2,j}}L_2(\theta_{2,j})=E_{((s,g),a,r,(s',g),v)\sim D_2}[(r+\gamma max_{a'}Q_2(s',a';\theta_{2,j-1},g)-Q_2(s,a;\theta_{2,j},g))\nabla_{\theta_{2,j}}Q_2(s,a;\theta_{2,j},g)] \tag{11}$$

状态-动作值函数的贝尔曼最优方程为:

$$\begin{aligned}Q_2^*(s,a;g)&=max_{\pi_{ag}}E[\sum_{t'=t}^{\infty}\gamma^{t'-t}r_{t'}\big|s_t=s,a_t=a,g_t=t,\pi_{ag}]\\&=max_{\pi_{ag}}E[r_t+\gamma max_{a_{t+1}}Q_2^*(s_{t+1}a_{t+1};g)\big|s_t=s,a_t=a,\\&\quad g_t=g,\pi_{ag}]\end{aligned} \tag{12}$$

# 4　基于势能的启发式奖赏塑形函数的分层强化学习算法

在本节中, 我们将介绍本文的主要贡献: 基于势能的启发式奖赏塑形函数的分层强化学习算法(Hierarchical Reinforcement Learning Algorithm of Heuristic Reward Shaping Function Based on Potential Energy, HRL-HRSF)。首先给出启发式信息的设计, 然后基于启发式信息提出了基于势能的奖赏塑形函数, 并证明了将该塑形函数引入基于分层 MDP 的渗透测试模型中不会改变其最优策略。

### 4.1 攻击路径发现启发式设计

为了提高智能体在渗透测试过程中的攻击路径发现效率, 重点在于能够将网络安全领域知识融入到路径规划中。因为在没有任何经验指导的情况下, 智能体需要采取随机策略来选取动作, 以试错的方式尽可能地获取具有正向奖赏地经验样本数据, 当智能体地动作空间较大时, 智能体在每一步决策时地可选空间变大, 在初始学习阶段, 智能体选择到可能带来正向奖赏值动作地可能性变小。因此, 通过引入领域知识, 利用启发式方法指导智能体更快地向目标状态移动, 有助于提高算法的收敛效率。然而不同启发式信息对于不同的动作选择侧重不同, 如果仅采用单一的启发式效果并一定能产生很好的效果, 如果利用多个启发式同时引导搜索, 则对于探索过程的指导性更强, 效果更明显。

考虑到在实际攻击过程中往往会出现以下两种现象: 1) 随着攻击者地不断渗透攻击, 目标网络中主机被入侵成功的数量是不断增长的; 2) 若目标网络中的主机数量较大, 则需要在每一步选择最有利于攻击者的主机, 以实现快速渗透到敏感主机的目的。因此, 本文根据攻击路径发现的经验知识, 提出基于深度横向渗透的启发式方法。从实际渗透测试角度出发, 攻击者在入侵目标主机之前, 最主要的目的就是进行跨网段的攻击, 实现子网间的快速横向渗透, 以达到尽快抵达目标主机的目的。基于此, 本文提出将网络中已入侵主机数量和已发现主机数量作为启发式信息构造奖赏函数, 通过给予跨网段攻击更多的奖赏值, 引导智能体在主机选择时更倾向于实现横向主机渗透, 从而实现大规模网络中的快速攻击路径发现。

(1) 已入侵主机数量

在基于分层 MDP 模型的渗透测试问题中, 只有成功入侵敏感主机才能获取奖赏, 在没有达到敏感主机之前, 智能体随机选择主机进行渗透, 即使攻击成功对于智能体前期探索而言没有正向奖赏。然而, 在实际渗透过程中, 实现入侵所有敏感主机之前, 每成功入侵一台中间主机, 对于智能体成功渗透敏感主机而言都可能有一定收益。因此, 本文将智能体在每成功攻击一台中间主机后给予一定的奖赏值, 通过辅助奖赏鼓励智能体在前期探索过程中用于启发式计算以促进智能体在前期训练过程中的探索效率, 加快学习进程。

(2) 已发现主机数量

在渗透测试问题中, 攻击者在成功渗透目标主机之前, 最主要的目标就是进行不同网段之间的渗透, 以更快地达到目标主机。以图 3 所示的渗透测试环境为例进行说明, 攻击者位于外网, 目标是子网 3 中的服务器(3,2), 防火墙限制了子网间的通信, 仅允许特定流量通过, 这里简化地用子网地址和主机地址表示主机, 则理想状态下的最佳攻击路径为: (1,0)→(2,0)→(3,2)。但是在训练过程中, 当攻击者在成功入侵子网 1 中的 Web 服务器后, 通过信息搜集、网络扫描等操作发现可攻击的目标为(1,1)、(2,0)和(2,1), 此时若攻击者选择主机(1,1)进行攻击, 即使攻击成功, 下一步可攻击的目标依然只能是(2,0)和(2,1), 对于渗透到目标主机没有任何价值, 反而增加了攻击成本, 所以最大化收益就是选择子网 2 中的主机进行攻击, 从而发现更多的可攻击目标, 更有利于实现目标主机的渗透。这里为了模拟真实渗透过程, 我们假设当攻击者成功渗透到一个新的子网中的主机, 就可以利用扫描操作获取到与该主机所在子网相连接的所有未发现过的子网主机, 以图 3 的网络环境为例, 当攻击者成功利用主机(1,0)作为跳板入侵到子网 2 中的主机(2,0), 此时就可以利用子网扫描操作发现子网 3 中的主机。因此, 本文将智能体成功实现跨网段的攻击给予更多的奖赏值, 以鼓励智能体在攻击路径发现过程中选择更有利于横向渗透的跨网段主机。

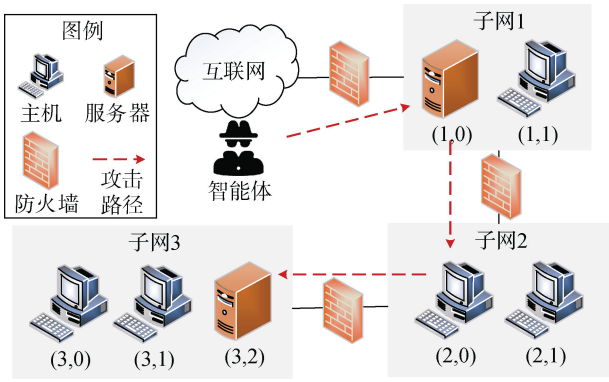


**图 4 渗透测试网络环境**

**Figure 4 Penetration testing network environment**

利用以上两种启发式信息, 我们给予智能体完成跨网段的横向渗透给予更多的奖赏值, 而对于同一子网内的攻击给予较少的奖赏值, 一方面可以鼓励智能体更倾向于实施深度横向渗透, 另一方面也可以鼓励智能体攻击更多的主机以快速达到目标主机。

### 4.2 基于势能的启发式奖赏塑形函数

基于以上思想, 本文提出一种基于势能的启发

式奖赏塑形函数, 就是将智能体当前在目标网络中已发现的主机数量和已入侵的主机数量作为当前时刻下的势能, 然后以此作为先验知识整合到攻击路径发现的分层强化学习算法中。我们将塑形函数应用到上层 MDP 中, 对于攻击路径发现过程来说, 上层主机间的渗透路径决定了下层单主机的攻击能否成功, 是智能体前期探索过程快慢的决定性因素, 因此, 本文通过对上层 MDP 的奖赏函数进行重新塑形, 利用主机选择过程中的附加奖赏值激励上层策略选择更有利于渗透攻击的主机, 加快了对未知环境的探索, 从而可以提高算法收敛速度。本文将势能函数定义如下:

**定义 1.**基于启发式信息的势能函数为:

$$\Phi(s)=\eta*[a(s)+b(s)] \tag{13}$$

其中 $\eta$ 是超参数, $a(s)$ 和 $b(s)$ 是两个函数, 分别用于计算在上层 MDP 的状态 $s$ 下已入侵和已发现的主机数量。

**定义 2.**基于势能的启发式奖赏塑形函数为:

$$F(s_t,g_t,s_{t+N})=\gamma\ \Phi(s_{t+N})-\Phi(s_t) \tag{14}$$

其中 $g_t$ 表示状态 $s_t$ 下选择的主机, $s_{t+N}$ 表示在状态 $s_t$ 下经过 $N$ 个时刻后转移的状态, 并且对于所有的 $s_{t+N}$, 均满足 $\Phi(s_{t+N})\geqslant\Phi(s_t)$。因为对于渗透测试过程, 攻击者在每一步采取攻击动作之后, 其发现和入侵的主机数量不会减少。

则塑形之后的上层 MDP 的奖赏函数为:

$$R=\sum_{h\in H}value(h)-\sum_{a\in A}cost(a)+F(s_t,a,s_{t+N}) \tag{15}$$

**定理 1.**给定本文中上层 MDP $M=(S,A,P,R)$ 和定义 2 中的基于势能的奖赏塑形函数, 转换为新的 MDP $M'=(S,A,P,R')$ 后, 其中 $R'=R+F$, 其最优策略不变。

证明. 对于 MDP $M'$, 塑形函数为 $F(s_t,g_t,s_{t+N})=\gamma\ \Phi(s_{t+N})-\Phi(s_t)$, 其中 $\Phi(s_{t+N})$ 与 $\Phi(s_t)$ 分别表示状态 $s_{t+N}$ 和 $s_t$ 下的势函数, Ng 等人[15]证明了任何利用基于状态的势函数对奖赏函数进行塑形, 都不会改变原有问题的最佳策略, 因此经过塑形后的上层 MDP $M'$, 利用塑形后的奖赏函数 $R'$ 进行求解得到的最优策略保持不变。

### 4.3 基于启发式奖赏塑形函数的分层强化学习算法

基于以上理论, 给出基于势能的启发式奖赏塑形函数的分层强化学习算法:

**算法 1.**基于势能的启发式奖赏塑形函数的分层强化学习算法.

输入: 经验池 $D_1$, $D_2$, 训练回合数 $N$, 每回合训练步数 $S$, 子目标训练步数 $T$, $Q_1(s,g;\theta_1)$ 参数 $\theta_1$, $Q_2(s,g,a;\theta_2)$ 参数 $\theta_2$, 上层 DQN 探索率 $\varepsilon_1$, 下层 DQN 探索率 $\varepsilon_2$, 超参数 $\eta$。

输出: $Q_1(s,g;\theta_1)$ 动作值函数, $Q_2(s,g,a;\theta_2)$ 动作值函数

1FOR *episode* = 1 to $N$ do:
2 初始化环境状态
3 FOR *steps* = 1 to $S$ do:
4 $g_t\leftarrow\varepsilon-Greddy(Q(s_t,G;\theta_1),\varepsilon_1)$
5 依据式(13)计算状态 $s_t$ 下的势能 $\Phi(s_t)$
6 FOR *goal_steps* = 1 to $T$ do:
7 $a_t\leftarrow\varepsilon-Greddy(Q(s_t,g_t,A;\theta_2),\varepsilon_2)$
8 执行动作 $a_t$, 获取上层奖赏 $f$ 以及观察下一状态 $s_{t+1}$
9 获取下层奖赏 $r'\leftarrow r(s_t,a_t,s_{t+1})$
10 存储序列 $((s_t,g_t),a_t,r',(s_{t+1},g_t),v)$ 到经验池 $D_2$
11 依据式(8)更新上层神经网络参数 $\theta_1$
12 依据式(11)更新下层神经网络参数 $\theta_2$
13 $R\leftarrow R+f$
14 end FOR
15 依据式(13)计算状态 $s_{t+1}$ 下的势能 $\Phi(s_{t+1})$
16 依据式(14)计算 $F(s_t,a,s_{t+1})$
17 依据式(15)计算奖赏 $R'$
18 存储序列 $(s_t,g_t,R',s_{t+1},V)$ 到经验池 $D_1$
19 end FOR
20 end FOR

## 5 实验分析

为了验证算法性能, 证明算法解决实际问题的能力, 我们在模拟网络攻击环境下进行测试, 实验使用 NASim[23]开源模拟网络环境平台来构建实验场景, NASim 是使用强化学习算法研究自动化渗透测试的基准平台, 其优点是开源、抽象和轻量, 并且可以方便的构建不同的网络场景, 使研究人员专注于提升算法性能。实验分为以下三个部分进行: 一是使用没有利用奖赏塑形的分层强化学习算法(HRL-None)及 HRL-HRSF 在同一个网络场景下进行测试, 以 DQN 算法和文献[10]中的 NDSPI-DQN 算法作为基准, 对比分析算法的收敛速度; 二是是构建不同规模的网络场景, 通过对比算法的收敛能力测试

HRL-None 算法、HRL-HRSF 算法和 NDSPI-DQN 算法的可扩展性; 三是考虑网络安全维护人员改变了目标网络环境的条件下, 使用预训练的 HRL-HRSF 算法模型, 测试算法的泛化性能, 算法的泛化性决定经过预训练的智能体是否可以实现未知环境下的快速攻击路径发现。

## 5.1 实验设置

为测试不同强化学习算法的收敛性能, 本文基于 NASim 提供的基准测试场景构建了不同的网络场景, 图 5 所示为实验场景 1。实验场景 1 是在 NASim 提供的基准测试场景上, 添加了蜜罐来增加网络的真实性和复杂性, 是一个典型的网络场景, 用以测试算法的收敛性能以及泛化性能。实验场景 2 到 4 为使用 NASim 的网络场景生成器自动生成的扩展网络场景, 其规模依次增大, 用以测试算法的可扩展性, 其中场景 2 和 3 保持漏洞数量不变, 扩展了网络中的主机数量, 场景 4 同时扩展了网络中的主机数量和漏洞(服务)数量。实验场景信息见表 3。

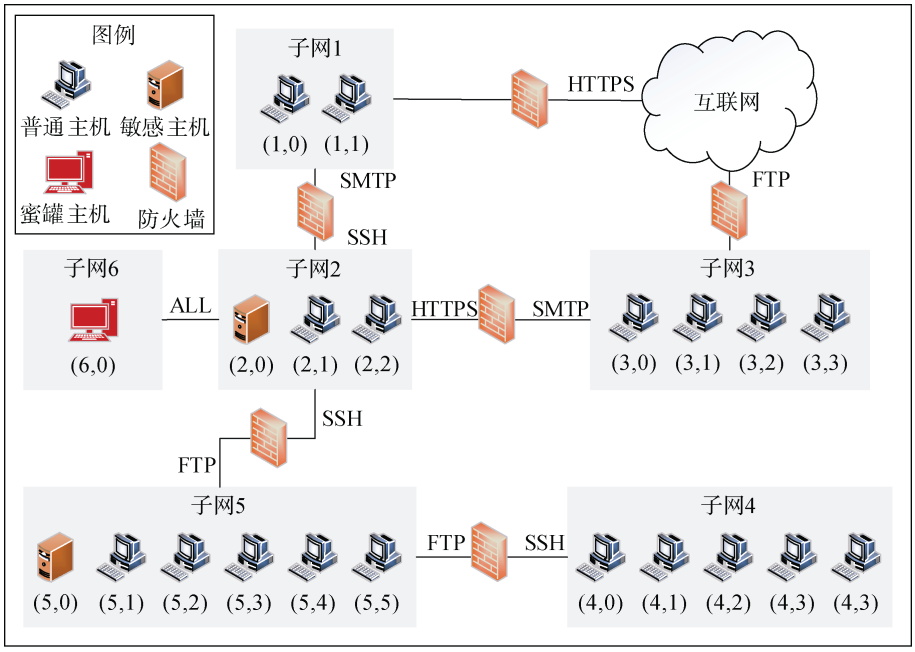


**图 5 实验场景 1 的网络拓扑图**

**Figure 5 Network topology of experimental scenario 1**

实验场景 1 中有 6 个子网, 21 台主机, 其中子网 1 和子网 3 与内外网连接, 子网 2、4、5 为内网, 子网 6 为蜜罐, 各子网之间边为防火墙访问控制规则, 防火墙控制规则为白名单, 决定子网之间允许通过的流量类型。网络中各个主机的详细配置信息见表 1, 每一台主机定义了地址(address)、操作系统(OS)、主机价值(value)、存在漏洞的服务(service)和进程(process), 主机的价值分别用 0, 100, –200 来表示普通主机、敏感主机和蜜罐主机。对每一台主机, 智能体可以采取表 2 中的操作执行, 实验选取内网渗透过程中经常被攻击者利用的服务和进程来指代漏洞, 如智能体可以利用 FTP 漏洞获取主机的 ROOT 权限, 利用 Tomcat 漏洞对获取到 USER 权限的主机进行提权。漏洞利用的概率在设置时使用的是 3.3 节介绍的方法。

为了测试算法在网络规模增大时的可扩展性, 本文使用 NASim 的场景生成器构建了 3 个规模不同的扩展网络场景, 见表 3 的场景 2 至场景 4。扩展的

**表 1 主机配置信息列表**

**Table 1 Host configuration information list**

| 地址 | 操作系统 | 价值 | 服务 | 进程 |
|---|---|---|---|---|
| (1,0), (1,1) | Linux | 0 | HTTP | Tomcat |
| (2,0) | Windows | 100 | SMTP | Schtask |
| (2,2) | Windows | 0 | SMTP | Schtask |
| (2,1), (3,0) | Windows | 0 | FTP | Schtask |
| (3,3) | Windows | 0 | FTP | Schtask |
| (3,1) | Windows | 0 | FTP, HTTP | Daclsvc |
| (3,2) | Windows | 0 | FTP | / |
| (4,0), (4,1), (5,2) | Linux | 0 | SAMBA | / |
| (4,3), (4,4) | Windows | 0 | SSH, FTP | Schtask |
| (5,0) | Windows | 100 | SMTP | Daclsvc |
| (4,2), (5,1) | Linux | 0 | SSH, HTTP | Tomcat |
| (5,3), (5,4), (5,5) | Linux | 0 | SSH | Tomcat |
| (6,0) | Windows | –200 | SSH, HTTP, FTP, SMTP, SAMBA | Schtask, Daclsvc |

表 2 智能体可执行操作列表

Table 2 Agent executable action list

| 操作 | 类别 | 操作系统 | 代价 | 概率 | 权限 |
|---|---|---|---|---|---|
| SSH-Exp | Exploit | Linux | 3 | 0.8 | User |
| FTP-Exp | Exploit | Windows | 1 | 0.5 | Root |
| HTTP-Exp | Exploit | None | 2 | 0.8 | User |
| SAMBA-Exp | Exploit | Linux | 2 | 0.2 | Root |
| SMTP-Exp | Exploit | Windows | 3 | 0.5 | User |
| Tomcat-PE | Promotion | Linux | 1 | 1 | Root |
| Daclsvc-PE | Promotion | Windows | 1 | 1 | Root |
| Schtask-PE | Promotion | Windows | 1 | 1 | Root |
| Subnet-Scan | Scan | / | 1 | 1 | / |
| OS-Scan | Scan | / | 1 | 1 | / |
| Service-Scan | Scan | / | 1 | 1 | / |
| Process-Scan | Scan | / | 1 | 1 | / |

网络场景中, 主机的配置和防火墙的访问控制规则是随机生成的, 网络中的敏感主机数量固定为 2, 敏感主机和普通主机的价值为 100 和 0, 拓展网络中没有蜜罐。在每一个场景中, 智能体的可执行操作数量为 Services + Processes + 4, 其中 4 表示子网、操作系统、服务和进程四种扫描操作。

表 3 实验网络场景列表

Table 3 List of experimental network scenarios

| 实验场景 | 主机 | 敏感主机 | 子网 | 服务 | 进程 | 蜜罐 |
|---|---|---|---|---|---|---|
| 场景 1 | 21 | 2 | 6 | 5 | 3 | Yes |
| 场景 2 | 100 | 2 | 21 | 5 | 3 | No |
| 场景 3 | 200 | 2 | 41 | 5 | 3 | No |
| 场景 4 | 50 | 2 | 11 | 80 | 5 | No |

在初始状态下, 智能体位于 Internet 中, 对目标网络的全局信息是未知的, 只能攻击与互联网连接的子网。代理需要根据扫描操作得到拓扑信息和主机配置信息, 同时选择一系列合适的动作进行横向移动。动作是指对某个主机采取的某种操作, 动作的有序序列就是攻击路径。在训练初始阶段, 智能体采取试错的方式进行动作选择, 每回合获取的累计奖励很小, 并且会陷入到蜜罐主机中, 随着训练的不断进行, 智能体逐渐学会了在避免入侵到蜜罐主机的情况下成功获取敏感主机的权限, 从而在每户和有限的步数内发现最佳攻击路径, 每个 Episode 结束的条件是:

- 获得所有敏感主机的 ROOT 权限;
- 训练步数达到设定的最大值;
- 陷入到蜜罐主机中。

## 5.2 实验参数设置

实验环境是基于 Python3.7 开发, 使用 Pytorch 作为算法代码框架, 硬件配置为 Intel Xeon Gold 6230R CPU, NVIDIA Quadro RTX6000 GPU, 开发及实验的操作系统为 Windows 10, 超参数设置如表 4 所示。

表 4 超参数设置

Table 4 Hyperparameter settings

| 超参数 | 含义 | 值 |
|---|---|---|
| Learning rate | 学习率 | 0.00025 |
| Batch size | 训练选取的样本批次大小 | 256 |
| Potential Hyper-parameter | 势能计算的超参数 | 100 |
| Discount factor | 折扣因子 | 0.9 |
| Hidden layer size | 隐藏层神经元层数及个数 | [128, 128] |
| Replay memory size | 经验回放池大小 | $D_1$=300000, $D_2$=50000 |
| Subgoal step limit | 子目标最大运行步数 | 100 |
| Episode step limit | 每回合最大运行步数 | 30000 |

## 5.3 实验结果分析

### 5.3.1 算法收敛性能对比分析

(1) 实验设计

DQN 算法已经被广泛应用到现有的自动化渗透测试威胁路径发现研究中[24-25], 同时文献[10]对 DQN 的改进算法很好的提高了攻击路径的发现效率。本实验以 DQN 和文献[10]中的 NDSPI-DQN 作为基准算法, 对比分析 DQN、NDSPI-DQN、HRL-None 以及 HRL-HRSF 算法在实验场景 1 中的收敛性能, 实验场景 1 规模适中, 且具有典型性。收敛性能的评价指标为算法在规定训练回合数上平均每回合的累积奖赏值的收敛情况, 以及在规定训练步数上平均奖赏值的收敛情况。

(2) 结果分析

图 6 所示为平均累积奖赏值(mean episode rewards)随训练回合数(training episodes)的变化, 图 7 为在规

定训练步数上(steps)平均奖赏值(mean rewards)的变化。智能体的学习目标是学会使用较少的步数获取目标网络中所有敏感主机的权限, 因此每回合累积奖赏值和平均奖赏值可以用来衡量智能体的策略水平。从图 6 和图 7 可以发现, 在训练的初始阶段, 智能体每回合所能获得的奖赏值较小, 每回合的探索步数达到设定的最大值, 但是随着训练的进行, 奖赏值不断增加, 说明智能体逐渐学会了用较少的步数来获取最大奖赏值的策略。最终, 除了 DQN 算法以外, NDSPI-DQN、HRL-None 和 HRL-HRSF 算法均能收敛到最大值累积奖赏值。其中, HRL-HRSF 算法性能最优, 可以在 200 回合以内收敛到最优值, HRL-None 次之, 在 300 回合内达到收敛, NDSPI-DQN 收敛速度明显差于前两种算法。在训练步数上, HRL-HRSF 算法要比 HRL-None 算法早 60000 步达到收敛, 比 NDSPI-DQN 算法早了 120000 步, 说明 HRL-HRSF 算法在每回合探索步数上要少于 NDSPI-DQN 和 HRL-None 算法, 也就导致 HRL-HRSF 算法训练时间要明显少于 NDSPI-DQN 和 HRL-None 算法。

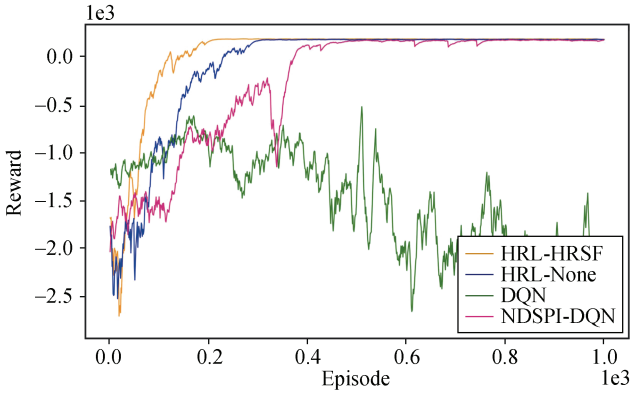


**图 6 场景 1 中平均累积奖励值随训练回合数变化**

**Figure 6 The average cumulative reward value in scenario 1 changes with the number of training rounds**

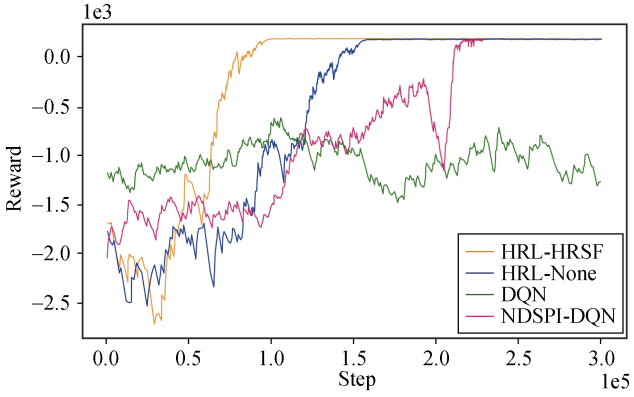


**图 7 场景 1 中平均奖励值随训练步数变化**

**Figure 7 In scenario 1, the average reward value changes with the number of training steps**

图 8 所示为场景 1 中 HRL-None 和 HRL-HRSF 算法在规定训练 1000 回合上最终选择的主机次数, 目标“2”、“7”、“14”、“15”分别表示主机(2,0)、(3,2)、(5,0)、(5,1), 智能体选择这些主机的次数远远大于其他主机, 说明这些主机构成了主机间渗透的最佳路径。在该策略下智能体首先利用主机(3,2)的 FTP 漏洞获取权限, 继而以(3,2)主机作为跳板, 对子网 2 进行扫描, 并利用 SMTP 漏洞对敏感主机(2,0)进行攻击, 然后继续以主机(2,0)为跳板, 对子网 5 进行扫描, 由于防火墙的设置, 智能体无法直接攻击主机(5,0), 因此先获取主机(5,1)的权限之后以此为跳板, 利用 SSH 漏洞获取了主机(5,0)的 ROOT 权限。至此智能体只需要攻击 4 台主机就可以获取所有敏感主机的权限, 所学策略与人工根据环境选择的最优策略一致。从结果来看, HRL-HRSF 算法与 HRL-None 算法主机选择结果一致, 没有改变 HRL-None 算法的最优策略, 说明该算法是有效的, 并且相较于 HRL-None 算法探索效率更高。

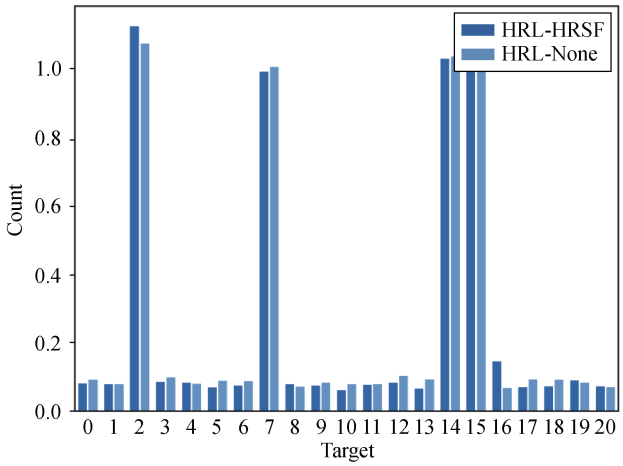


**图 8 实验场景 1 中的主机选择次数**

**Figure 8 Host selection times in experiment scenario 1**

在收敛性能上, NDSPI-DQN、HRL-HRSF 和 HRL-None 算法均能有效收敛到最优值, 在收敛速度上 HRL-HRSF 最优, HRL-None 次之, NDSPI-DQN 最差。相比之下, DQN 算法在经过震荡过后未能在设定的回合数内收敛到最优值。这是因为在强化学习任务中, 奖赏是智能体学习的依据, 起到监督信号的作用, 稀疏的奖赏环境中无法及时获取到反馈信号会导致强化学习算法迭代缓慢。此外, 蜜罐网络的加入给实验场景 1 增加了复杂性, 负奖赏值的存在会干扰智能体的探索过程, 使得智能体越发难以探索到原本就稀疏的正向奖赏。HRL-None 和 HRL-HRSF 相对于 DQN 和 NDSPI-DQN 而言, 通过将攻击路径

发现过程建模为分层的 MDP 模型, 大大减少了智能体在训练过程中的状态空间和动作空间, 使得收敛性能得到优先提升。同时, HRL-HRSF 算法相比于 DQN、NDSPI-DQN 和 HRL-None 算法而言, 通过合理的奖赏机制, 使智能体的学习能力有了显著提升, 使得算法在处理复杂场景问题时, 在奖赏稀疏的条件下能够充分的探索和试错寻找到最优策略, 从而可以利用更少的学习步数达到最大累积奖赏值。

### 5.3.2 算法可扩展性测试及分析

(1) 实验设计

当网络规模增大时, 智能体受奖赏稀疏问题的影响也就越大, 在真实的渗透测试任务中, 测试人员往往会面对规模庞大的网络环境, 并且可供选择的漏洞范围不定。为比较算法的可扩展性, 验证网络规模变化对算法收敛性能的影响, 本实验构建了三个不同规模的网络测试场景(见表 3 中的场景 2 至场景 4), 其中场景 2 至场景 3 为控制漏洞数量不变, 只改变主机数量; 场景 4 不仅改变主机数量, 而且增加漏洞数量。实验进行对比的算法是 NDSPI-DQN、HRL-None 和 HRL-HRSF, 由于在实验场景 1(21 台主机)中 DQN 算法无法收敛, 所以在大规模网络场景下对比可扩展性时没有将其作为基准对比算法。算法可扩展性能的评判指标是智能体的学习效率, 即平均累积奖赏值随训练步数的变化情况。

(2) 结果分析

图 9、图 10 所示为使用 NDSPI-DQN、HRL-None 和 HRL-HRSF 算法训练的智能体在改变主机数量的网络场景下, 平均累积奖赏值随训练步数的变化情况, 其中图 9 为实验场景 2(100 台主机)的实验结果, 图 10 为实验场景 3(200 台主机)的实验结果。从上述图中可以看到, 随着网络规模增大, NDSPI-DQN 受网络规模的影响最大, HRL-None 算法次之, 并且在主机数量达到 200 台主机时, 这两种算法都无法有效收敛。相比之下, HRL-HRSF 算法效果最好, 有较好的鲁棒性, 在两个场景下其平均累积奖赏值均能有效收敛到最优值, 并且从收敛速度来看, 随着主机数量的增多, HRL-HRSF 算法的优势越明显, 其收敛所需要的训练步数相比于 NDSPI-DQN 和 HRL-None 算法越来越少, 说明在探索过程中给予更多的正向奖赏, 有利于智能体快速学习到更有价值的攻击路径。

图 11 是 NDSPI-DQN、HRL-None 和 HRL-HRSF 算法在改变网络中改变主机数量和漏洞数量的情况下, 平均累积奖赏值随训练步数的变化。从图 11 中可以看到, 三种算法在 50 台主机、80 个漏洞的场景中均能有效收敛, 相比之下, HRL-HRSF 收敛速度最快, NDSPI-DQN 效果最差, 鲁棒性不如前两种算法。

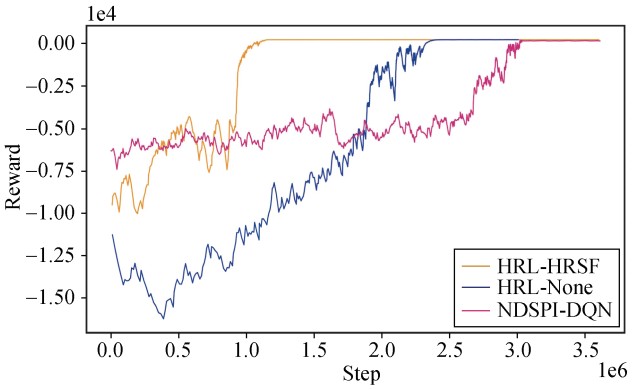


**图 9 场景 2(100 台主机)中平均奖励值随训练步数变化**

**Figure 9 The average reward value in scenario 2 (100 hosts) varies with the number of training steps**

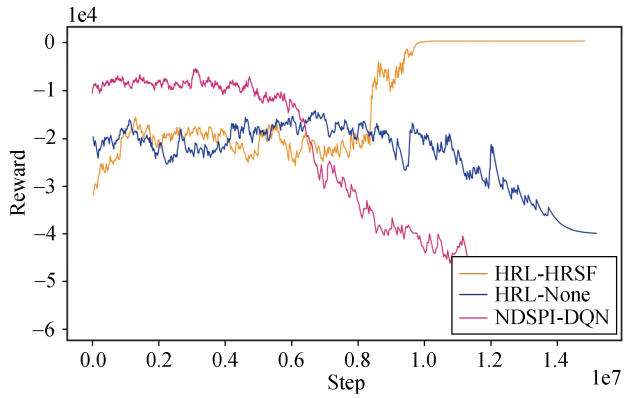


**图 10 场景 3(200 台主机)中平均奖励值随训练步数变化**

**Figure 10 The average reward value in scenario 3 (200 hosts) varies with the number of training steps**

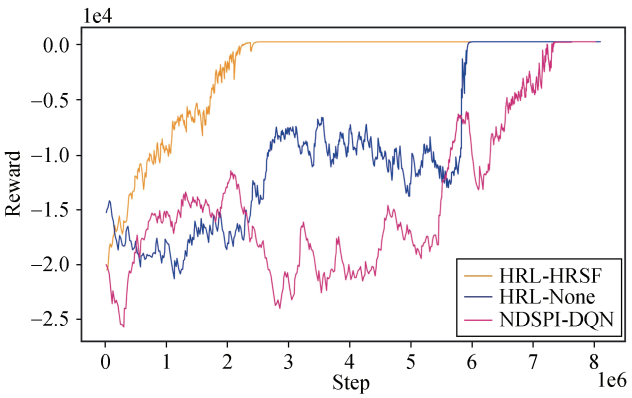


**图 11 场景 4(50 台主机, 网络中漏洞数量为 80)中平均奖励值随训练步数变化**

**Figure 11 The average reward value in scenario 4 (50 hosts, 80 vulnerabilities in the network) changes with the number of training steps**

### 5.3.3 算法泛化性能分析

(1) 实验设计

渗透测试过程具有动态性特点, 在现实世界中, 网络维护人员在检测到攻击流量时, 往往会采取缓解措施改变网络环境, 算法的泛化能力好坏决定着预训练的智能体在面对陌生环境时能否做到快速的路径发现。

为测试算法的泛化能力, 本文在实验场景 1 的基础上, 基于现实中网络维护人员在发现黑客入侵之后可能采取的行为, 通过改变主机配置信息以及网络拓扑的方式构建了两种测试场景(见表 5)。其中测试场景 1 假想的是网络维护人员发现子网 3 有黑客入侵痕迹, 遂修复了子网 1 中主机的漏洞; 测试场景 2 是网络维护人员发现主机(2,0)有黑客入侵痕迹, 在修复子网 3 中主机漏洞的同时, 遂将子网 2 中的敏感主机转移到了子网 4。

**表 5 基于实验场景 1 的泛化能力测试场景**

**Table 5 Generalization ability test scenario based on experimental scenario 1**

| 测试场景 | 网络维护人员措施 | 改变形式 |
|---|---|---|
| 测试场景 1 | 修复子网 3 中所有主机的漏洞 | 改变主机配置 |
| 测试场景 2 | 在修复子网 3 中漏洞的同时, 将敏感主机(2,0)转移到子网 4 | 改变网络拓扑 |

实验首先在实验场景 1 中使用 HRL-HRSF 算法对智能体进行预训练, 之后分别在两个测试场景上进行实验, 记录预训练智能体累积奖赏值收敛情况。与之相对比的是未经预训练直接在测试场景上使用 HRL-HRSF 算法进行训练的智能体。

(2) 结果分析

图 12 所示为测试场景 1 下使用预训练智能体和不使用预训练智能体时算法的收敛情况, 图 13 所示为测试场景 2 下使用预训练智能体和不使用预训练智能体时算法的收敛情况。从图 12 中可以看到, 在网络维护人员修复网络中部分主机漏洞之后, 没有预训练的智能体需要大约 150000 步才能学习到最优策略, 而经过预训练的智能体可以在 50000 步以内学习到最优策略。更进一步, 图 13 说明网络中部分拓扑发生改变时, 经过预训练的智能体也可以在 80000 步以内实现更快收敛。这说明在考虑了网络维护人员缓解措施时, 使用 HRL-HRSF 算法预训练的智能体可以做到“举一反三”, 面对改变的环境可以做到快速的攻击路径发现, 算法具有较好的泛化能力。

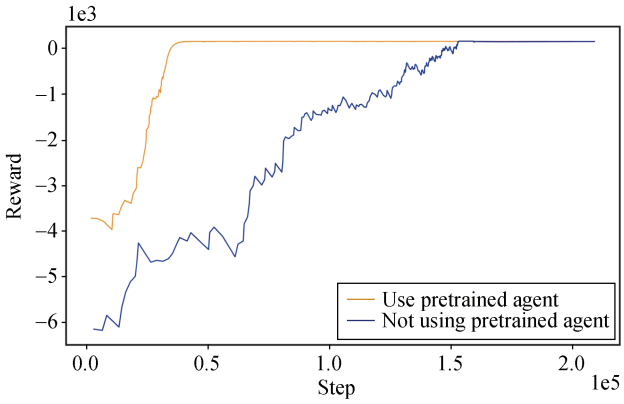


**图 12 测试场景 1 中平均奖励值随训练步数变化**

**Figure 12 The average reward value in test scenario 1 changes with the number of training steps**

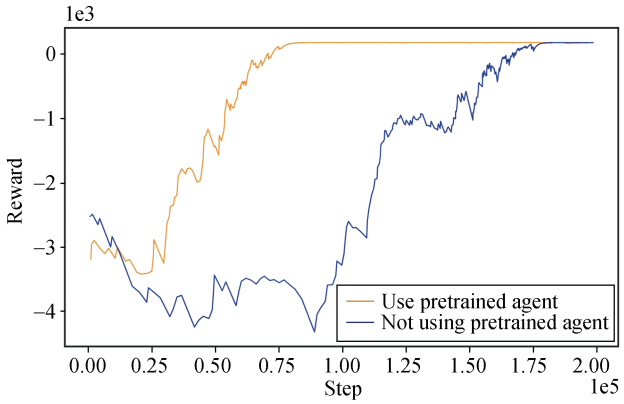


**图 13 测试场景 2 中平均奖励值随训练步数变化**

**Figure 13 The average reward value in test scenario 2 changes with the number of training steps**

## 6 结束语

在强化学习中, 延迟奖赏或稀疏奖赏通常会导致学习效率低下, 基于强化学习的智能化攻击路径发现技术同样存在这些问题, 导致其在大规模网络环境中出现探索效率低、收敛速度慢等问题。而在许多实际环境中, 已经有很多研究利用领域知识设计辅助奖赏函数, 为智能体的训练提供更多的指导信息。基于此, 本文提出了一种基于势能的启发式奖赏塑形函数的分层强化学习算法(HRL-HRSF), 首先基于分层的 MDP 模型对攻击路径发现过程进行建模, 并基于该模型设计了一种分层强化学习算法, 然后以该算法为基础, 利用渗透测试中的先验知识构造出基于启发式信息的势能函数, 最后利用该势能函数构造奖赏塑形函数并应用到分层强化学习算法中, 对原有奖赏函数进行重新塑形, 有效解决了智能体稀疏奖赏的问题。实验结果表明, 与没有经过奖赏塑形的分层强化学习算法、DQN 及其改进算法相比, HRL-HRSF 算法的收敛性能最好, 同时在大规模网

络中依然保持良好的鲁棒性。受限于强化学习与环境的强交互性，当前应用强化学习的渗透测试仍处于模拟验证阶段，如何构建更符合真实渗透测试过程的模拟网络场景，以及应用虚拟化技术实现在仿真网络甚至真实网络中的渗透测试攻击路径发现是未来的研究方向。

## 参考文献


[1] Arce I, McGraw G. Guest Editors' Introduction: Why Attacking Systems is a Good Idea[J]. *IEEE Security & Privacy*, 2004, 2(4): 17-19.

[2] Arkin B, Stender S, McGraw G. Software Penetration Testing[J]. *IEEE Security & Privacy*, 2005, 3(1): 84-87.

[3] Wiewiora E, Cottrell GW, Elkan C. Principled methods for advising reinforcement learning agents[C]. *In: Proc. of the 20th Int'l Conf. on Machine Learning. Menlo Park: AAAI Press*, 2003: 792-799.

[4] Barto A G, Mahadevan S. Recent Advances in Hierarchical Reinforcement Learning[J]. *Discrete Event Dynamic Systems*, 2003, 13(1): 41-77.

[5] Silver D, Huang A, Maddison C J, et al. Mastering the Game of Go with Deep Neural Networks and Tree Search[J]. *Nature*, 2016, 529: 484-489.

[6] Vinyals O, Babuschkin I, Czarnecki W M, et al. Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning[J]. *Nature*, 2019, 575(7782): 350-354.

[7] Sarraute C, Buffet O, Hoffmann J. Penetration Testing == POMDP Solving? [EB/OL]. 2013: arXiv: 1306.4714. http://arxiv.org/abs/1306.4714.pdf.

[8] Zhou T Y, Zang Y C, Zhu J H, et al. NIG-AP: A New Method for Automated Penetration Testing[J]. *Frontiers of Information Technology & Electronic Engineering*, 2019, 20(9): 1277-1288.

[9] Zhou S C, Liu J J, Zhong X F, et al. Intelligent Penetration Testing Path Discovery Based on Deep Reinforcement Learning[J]. *Computer Science*, 2021, 48(7): 40-46.
(周仕承, 刘京菊, 钟晓峰, 等. 基于深度强化学习的智能化渗透测试路径发现[J]. *计算机科学*, 2021, 48(7): 40-46.)

[10] Zhou S C, Liu J J, Hou D D, et al. Autonomous Penetration Testing Based on Improved Deep Q-Network[J]. *Applied Sciences*, 2021, 11(19): 8823.

[11] Babes M, de Cote E M, Littman M L. Social Reward Shaping in the Prisoner's Dilemma[C]. *The 7th international joint conference on Autonomous agents and multiagent systems - Volume 3*, 2008: 1389-1392.

[12] Marthi B. Automatic Shaping and Decomposition of Reward Functions[C]. *The 24th international conference on Machine learning*, 2007: 601-608.

[13] Hu T R, Zang Y C, Cao R R, et al. Research on Attack Path Discovery Algorithm Based on Multi-Heuristic Information Fusion[J]. *Journal of Cyber Security*, 2021, 6(3): 202-211.
(胡泰然, 臧艺超, 曹蓉蓉, 等. 基于多启发式信息融合的攻击路径发现算法研究[J]. *信息安全学报*, 2021, 6(3): 202-211.)

[14] Ng AY, Russell SJ. Algorithms for inverse reinforcement learning[C]. *In: Proc. of the 17th Int'l Conf. on Machine Learning. New York: Morgan Kaufmann Publishers*, 2000: 663-670.

[15] Ng A Y, Harada D, Russell S J. Policy Invariance under Reward Transformations: Theory and Application to Reward Shaping[C]. *The Sixteenth International Conference on Machine Learning*, 1999: 278-287.

[16] Marom O, Rosman B. Belief Reward Shaping in Reinforcement Learning[J]. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018, 32(1): 3762-3769.

[17] Devlin S, Kudenko D. Dynamic Potential-Based Reward Shaping[C]. *The 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, 2012: 433-440.

[18] Y Gao and F Toni. Potential based reward shaping for hierarchical reinforcement learning[C]. *In Proc. 24th Int. Conf. Artif. Intell. Menlo Park, CA, USA: AAAI Press*, 2015: 3504-3510.

[19] Chen T L, Lu J. Towards Analysis of Semi-Markov Decision Processes[C]. *The 2010 international conference on Artificial intelligence and computational intelligence: Part I*, 2010: 41-48.

[20] Mahadevan S, Marchalleck N, Das T K, et al. Self-Improving Factory Simulation Using Continuous-Time Average-Reward Reinforcement Learning[J]. *Proc 14th International Conference on Machine Learning*, 1997(August): 202-210.

[21] Efthymiadis K, Kudenko D. Using Plan-Based Reward Shaping to Learn Strategies in StarCraft: Broodwar[C]. *2013 IEEE Conference on Computational Inteligence in Games*, 2013: 1-8.

[22] Backes M, Hoffmann J, Kvnnwmann R, et al. Simulated penetration testing and mitigation analysis[EB/OL]. 2017: ArXiv: abs/1705.05088.

[23] Schwartz J, Kurniawatti H. NASim: Network Attack Simulator[Z]. https://networkattacksimulator.readthedocs.io/. 2022.

[24] Chowdhary A, Huang D J, Mahendran J S, et al. Autonomous Security Analysis and Penetration Testing[C]. *2020 16th International Conference on Mobility, Sensing and Networking*, 2020: 508-515.

[25] Schwartz J, Kurniawati H. Autonomous Penetration Testing Using Reinforcement Learning[EB/OL]. 2019: arXiv: 1905.05965. http://arxiv.org/abs/1905.05965.pdf.

**曾庆伟** 于 2017 年在南京邮电大学信息安全专业获得学士学位。现在陆军工程大学网络空间安全专业攻读硕士学位。研究领域为网络安全。研究兴趣包括: 渗透测试、强化学习。Email: 943919527@qq.com

**张国敏** 于 2009 年在解放军理工大学计算机科学与技术专业获得博士学位。现任陆军工程大学副教授。研究领域为网络安全和网络管理。研究兴趣包括: 网络测量、网络主动防御、网络渗透测试。Email: zhang_gmwn@163.com

**邢长友** 于 2009 年在解放军理工大学计算机科学与技术获得博士学位, 现任陆军工程大学教授。研究领域为网络安全, 研究兴趣包括: 网络空间测绘与对抗、网络主动防御、网络功能虚拟化。Email: changyouxing@aeu.edu.cn

**宋丽华** 于 2006 年在解放军理工大学通信与信息系统专业获得博士学位, 现任陆军工程大学教授。研究领域为计算机网络, 研究兴趣包括: 网络主动防御。Email: songlihua_mail@189.cn