计算机科学
COMPUTER SCIENCE

基于分层强化学习的智能化攻击路径发现方法

曾庆伟, 张国敏, 邢长友, 宋丽华

引用本文

曾庆伟, 张国敏, 邢长友, 宋丽华.基于分层强化学习的智能化攻击路径发现方法[J]. 计算机科学, 2023, 50(7): 308-316.

ZENG Qingwei, ZHANG Guomin, XING Changyou, SONG Lihua. Intelligent Attack Path Discovery Based on Hierarchical Reinforcement Learning [J]. Computer Science, 2023, 50(7): 308-316.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220500101

基于分层强化学习的智能化攻击路径发现方法

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 210007
(943919527@qq.com)

摘　要　智能化攻击路径发现是开展自动化渗透测试的一项关键技术,但现有方法面临着状态、动作空间呈指数型增长和奖励稀疏等问题,导致算法难以收敛。为此,提出了一种基于分层强化学习的智能化攻击路径发现方法 iPathD(Intelligent Path Discovery)。iPathD 将攻击路径发现过程构建为一个分层的马尔可夫决策过程,以分别描述上层的主机间渗透路径发现和下层的单主机内部攻击路径发现,并在此基础上提出并实现了一种基于分层强化学习的攻击路径发现算法。实验结果表明,与传统基于 DQN(Deep Q Learning)及其改进算法的方法相比,iPathD 路径发现方法更加快速有效,并且随着主机中漏洞数目的增加,iPathD 的效果更好,且适用于大规模的网络场景。

关键词:渗透测试;马尔可夫决策过程;分层强化学习;攻击路径发现;DQN 算法
中图法分类号　TP393

Intelligent Attack Path Discovery Based on Hierarchical Reinforcement Learning

ZENG Qingwei,ZHANG Guomin,XING Changyou and SONG Lihua
College of Command and Control Engineering,Army Engineering University,Nanjing 210007,China

Abstract　Intelligent attack path discovery is a key technology for automated penetration testing,but existing methods face the problems of exponential growth of state and action space and sparse rewards,which make the algorithm difficult to converge. To this end,an intelligent attack path discovery method(iPathD) based on hierarchical reinforcement learning is proposed. iPathD constructs the attack path discovery process as a layered Markov decision process to describe the upper-layer inter-host penetration path discovery and the lower-layer single-host internal attack path discovery,respectively. On this basis,an attack path discovery algorithm based on hierarchical reinforcement learning is proposed and implemented. Experimental results show that compared with the traditional method based on deep Q learning(DQN) and its improved algorithm,the iPathD path discovery method is faster and more effective. With the increase of the number of vulnerabilities in the host,the effect of iPathD is better,and it is suitable for large-scale network scenarios.

Keywords　Penetration testing,Markov decision process,Hierarchical reinforcement learning,Attack path discovery,DQN algorithm

1　引言

渗透测试是通过模拟黑客攻击,在不影响目标系统网络的前提下,利用系统漏洞来获得系统控制权的网络安全测试方法[1-2]。通过该方法,目标系统即可根据报告中的漏洞和脆弱点,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。传统的渗透测试主要依赖人工方式进行,然而现代企业和组织的网络系统庞大而复杂,使得单纯依赖人工手段来实施安全测试不切实际,需要耗费大量的时间成本和人力成本,增大了渗透测试的代价。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,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)是一种学习范式,其通过探索和利用未知环境来获取经验知识,并根据经验学习在环境中采取行动的最佳策略,可以很好地解决序贯决策问题[3]。与监督学习和无监督学习相比,强化学习的优势在于不依赖大量的静态数据,可以在没有或者有限先验知识的情况下,根据经验学习最大奖励值策略,因此特别适用于实时和对抗性的环境。渗透测试就是一种与环境实时交互的动态决策过程,测试人员通过采取各种攻击动作,根据网络环境给予的反馈,来发现主机的漏洞和配置信息,从而通过分析找到网络中的脆弱点。因此,强化学习技术为实现智能化攻击路径发现提供了新思路。

到稿日期:2022-05-12　返修日期:2022-08-18
基金项目:国家自然科学基金面上项目(62172432)
This work was supported by the National Natural Science Foundation of China(62172432).
通信作者:张国敏(40519667@qq.com)

近年来，基于强化学习的攻击路径发现相关研究成为了相关领域的热点。但是目前还面临着很多挑战：1）随着网络规模的不断增大，网络中的主机不断增多，智能体的状态和动作空间呈指数型增长；2）大规模网络环境下稀疏奖励的问题愈发严重。因此，如何解决智能化攻击路径发现过程中的这些问题，提高智能体在大规模网络环境下攻击路径的发现效率，是本文的研究重点。本文的主要贡献如下：

（1）为了解决上述问题，本文基于分层的思想将攻击路径发现过程构建为一个分层的马尔可夫决策过程，其中，上层策略进行主机间的渗透路径发现，下层策略进行单主机的攻击路径发现，通过将渗透测试连续性的状态变化过程分解为间歇性状态变化过程，将渗透测试问题分解为一个个子任务，使得智能体状态、动作空间爆炸和稀疏奖励的问题得到有效缓解，从而提高训练效率。

（2）针对分层马尔可夫决策过程，提出了一种基于分层强化学习的攻击路径发现算法，该算法利用两层神经网络，分别完成主机选择和攻击动作选择，实现了攻击路径的快速发现。

（3）将基于开源的网络攻击模拟器 NASim 构建的网络场景用于智能体训练，通过对比实验，验证了本文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。

2 相关工作

在人工智能领域中，最初将人工智能与渗透测试相结合是利用部分可观测马尔可夫决策模型（Partially Observable Markov Decision Process，POMDP）来描述智能体在不确定性环境下的决策方式，因为 POMDP 可以很好地建模真实渗透测试场景下攻击者对目标环境的不确定性。Sarraute 等[4]将渗透测试建模为 POMDP 问题，将渗透测试过程中的信息收集阶段纳入攻击路径生成过程，首次提供了将扫描操作与实际漏洞利用操作智能混合的方法，实现了对单个主机的自动化渗透测试。Shmaryahu 等[5]将渗透测试建模为部分可观察的偶发问题，并设计了偶发规划树算法来规划攻击路径。上述求解方法只能进行单机攻击路径发现。为了能够实现网络层面的自动化渗透测试，Sarraute 等[6]提出了 4AL 算法，该方法主要包含 3 个阶段：首先将目标网络分解为有向无环图，然后对图中全连通节点进行分解，将属于同一个子网的节点划分到一起，利用 POMDP 模型对每一个节点进行攻击规划，最后整合形成攻击路径。基于 POMDP 的方法虽然能很好地建模渗透测试过程中的不确定性，但是该方法的计算复杂性随着问题规模呈指数增长，一般只适用于求解一些小规模问题，仍然无法很好地解决大规模网络场景下的攻击路径发现问题。

基于强化学习的方法通常将渗透测试过程形式化为马尔可夫决策过程（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。Zennaro 等[7]将 Q-learning 应用到网络安全夺旗赛（CTF）中，实现了强化学习在简单渗透测试场景下的验证。Zhou 等[8]将信息熵引入强化学习的奖励值设计，提出了一种基于网络信息增益的攻击路径发现算法，该方法利用网络信息获取奖励，引导智能体选择最佳动作。为了提高自动化渗透测试在实际生活中的应用，Hu 等[9]提出了一种自动化渗透测试框架，该框架首先利用 Shodan 搜索引擎收集目标网络数据，以构建一个真实的网络拓扑，并使用 MulVAL 为该拓扑生成攻击图，最后利用 DQN 算法发现最优攻击路径。Zhou 等[10]提出了一种改进的 DQN 算法——NDSPI-DQN，通过整合 5 种 DQN 的扩展，包括噪声网络、Soft Q-learning、Dueling 结构、优先经验回放和内在好奇心模块，来提高智能体在大规模场景中的探索效率。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，Schwartz 以及微软安全团队分别设计了 NASim[11]以及 CyberBattleSim[12]平台，提供了可以用于强化学习训练的模拟测试基准平台，以测试不同算法的性能。

基于 MDP 的上述研究中，文献[7-9]仅实现了在简单网络场景下的实验验证，文献[10]通过改进的 DQN 算法虽然提高了算法的可扩展性，实现了较大规模网络场景下的攻击路径发现，但是仍然没有解决状态、动作空间高度离散和稀疏奖励等问题，在算法收敛效率和可扩展性上还有待提高。

3 问题分析与算法理论

3.1 问题描述

攻击路径发现是一种典型的序贯决策问题，目标就是在每个时间状态下，通过观测网络环境状态、执行动作来自动化完成渗透测试过程。在这个过程中，智能体用环境给予的奖励值来衡量当前动作的优劣，进而调整策略，找到最优攻击路径。图 1 给出了典型场景下智能体进行渗透测试的攻击路径示例，智能体通过执行各类动作在各个子网中进行横向移动，最终获取内网敏感主机的最高权限，攻击路径就是智能体横向移动过程中执行的动作序列$\vec{a}=(a_{1,i},a_{2,j},a_{3,k})$。

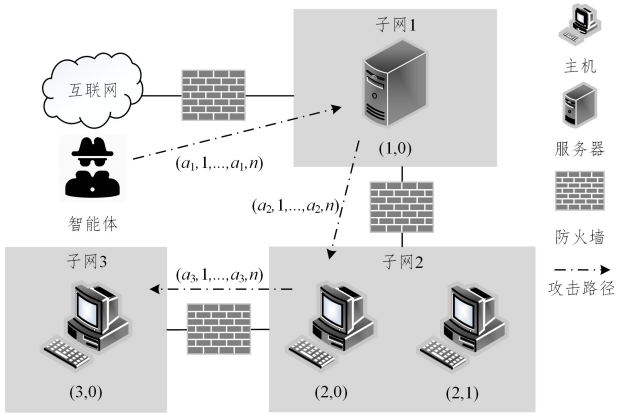


图 1 攻击路径示例

Fig. 1 Example of attack path

马尔可夫决策过程（Markov Decision Process，MDP）是建模离散决策问题的通用框架，可以很好地模拟渗透测试过程，文献[10,13]将渗透测试过程建模为 MDP 模型，用四元组$\langle S,A,R,P\rangle$表示，其中状态空间 S 表示智能体观测到的网络环境状态信息，包括智能体在渗透过程中的位置和已知的网络拓扑信息、主机配置信息、漏洞信息等；动作空间 A 表示智能体可以对目标主机采取的所有动作，如扫描、漏洞利用或

者权限提升等，智能体动作的执行会改变环境状态，同时获取环境反馈的奖励值；奖励函数 R 表示智能体执行动作后从外部环境中获得的奖励值，奖励值是对当前动作好坏的反馈，智能体学习的目标是最大化累积奖励值；状态转移概率 P 与主机配置、防火墙设置、漏洞利用成功概率等有关。当把渗透测试问题形式化为 MDP 问题时，就可以用强化学习算法来解决攻击路径发现问题。

根据以上的建模方式，可以形式化地描述渗透测试的过程，但是随着网络规模的增大，也暴露出了以下问题。

(1)状态空间爆炸问题。在 MDP 中，状态空间通常用向量表示，向量的维度则与主机数量和主机配置信息有关。对于一个有 M 台主机的网络，每台主机的漏洞有 N 个，状态空间大小为 $O(3^{M\times N})$，3 表示每个漏洞有 3 种状态：漏洞存在、不存在或者未知。因此，随着网络规模的增大，状态空间呈指数级增长，严重限制了强化学习算法的收敛效率。

(2)大规模动作空间问题。渗透测试的动作可以表示为向量 $\boldsymbol{a}=\langle h,o\rangle$，其中 h 和 o 分别代表主机地址和操作类别，那么对于一个具有 M 台主机和 N 个攻击动作的网络来说，动作空间大小为 $O(M\times N)$。对于智能体而言，因为无法得知每台主机上漏洞存在的种类及数量，在动作选择时对每一台主机都会选择所有可能的漏洞利用操作，所以随着网络规模的增大，漏洞的种类和数量也会增加，导致强化学习智能体的动作空间增大，使得智能体在每一步决策时的可选空间变大，路径搜索变得越来越困难，造成算法难以收敛。

(3)稀疏奖励问题。在渗透测试过程中，智能体只有成功获取敏感主机的权限才能获取正向收益，因此在网络规模不断增大的情况下，稀疏奖励问题也越发突出，算法同样难以收敛。

因此，为了解决以上问题，提升智能体的学习效率，我们将渗透测试过程分解为主机选择和动作选择的分层结构，然后利用分层强化学习算法实现攻击路径的快速发现。

3.2 分层强化学习

分层强化学习(Hierarchical Deep 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[14]。分层是通过抽象实现的，实现 HRL 的抽象技术主要包括状态抽象[15]、时态抽象[16]、状态空间分解[17]这 3 种方法。无论使用哪种方法，都是以半马尔可夫决策过程(Semi-Markov Decision Process，SMDP)[18]为模型基础，即通过将 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 2 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 细分为连续时间离散事件 SMDP[19]和离散时间 SMDP[18]。

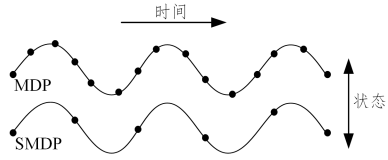


图 2 MDP 与 SMDP

Fig. 2 MDP and SMDP

在 SMDP 中，设 τ 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，奖励值为 $R(s,a)$。则状态值函数和状态-动作值函数 Bellman 的最优方程为：

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

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

4 模型与结构

4.1 分层设计

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，不仅包括单主机的攻击过程，还包括主机之间的渗透路径。智能体在对单主机的攻击过程中，其相对于整个网络主机状态而言在一定时间内没有变化，只有成功攻击该主机后才会攻击其他主机，状态才会发生改变。因此，我们将攻击路径发现过程分解为主机间横向渗透过程和单主机渗透攻击过程，具体的攻击路径发现过程如图 3 所示。上层策略中节点表示主机的位置信息，包括子网编号和主机编号；下层策略表示单主机的攻击过程，包括扫描、漏洞利用和提权等动作，矩形框中的内容就是某时刻的攻击动作。

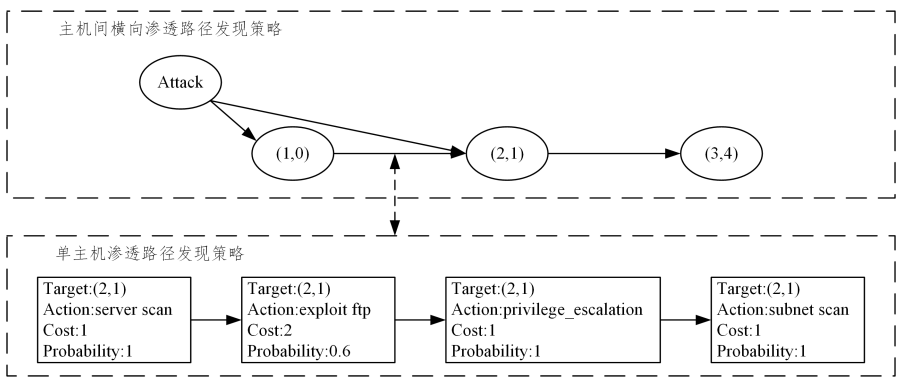


图 3 基于分层设计的攻击路径发现示例

Fig. 3 Example of attack path discovery based on hierarchical design

(1)智能体主机间横向渗透路径发现策略。上层策略表示主机间的宏观渗透关系，在实际网络渗透过程中，虽然攻击人员会在内网不断渗透攻击，控制一台又一台主机，但是其在单主机渗透过程中的状态变化较为缓慢，攻击位置在一段时间内保持不变，因此上层策略仅表示目标网络中各主机是否已成功入侵，它可以表示渗透过程中逐个攻击内网主机的过程，这个层面的目的就是找到一条从攻击者的主机到目标主机的一条尽可能短的路径。

(2)智能体单主机攻击路径发现策略。下层策略表示的是任意一台主机的微观渗透过程，在上层策略选择了下一步攻击的目标主机后，下层策略就根据可选择的攻击动作进行单主机的攻击过程，规划具体的攻击动作序列，包括主机扫描、漏洞利用和系统提权等动作。

4.2 渗透测试网络模型

基于以上思想，我们提出了一种基于分层强化学习的智能化攻击路径发现方法 iPathD。首先将渗透测试问题用分层的马尔可夫决策过程进行建模，利用一种新型的状态表征方法来处理攻击路径发现中的大型状态空间，并将攻击动作分解为主机选择和攻击动作选择，以减少大规模动作空间问题，通过引入中间奖励值来促进智能体的学习动力，从而缓解稀疏奖励问题，最后利用分层强化学习算法进行求解。

分层的马尔可夫决策过程分别用四元组$\{S,A,R,T\}$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励，T为状态转移函数。对于上层的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，对于一个有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。例如，在某时刻图1中的主机(1,0)和主机(2,1)被控制，则受控主机状态向量为(1,0,1,0)。动作空间A定义为选择下一步攻击的目标主机，对于一个有M台主机的网络，其动作空间大小为$O(M)$。奖励函数R决定了能否找到一条主机之间的最佳渗透路径，我们将奖励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，将不同的动作(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，奖励就是执行操作的代价。如式(3)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。

$$R=\sum_{h\in H}value(h)-\sum_{a\in A}cost(a) \tag{3}$$

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，如图4所示。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的种类数，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的所有主机总数，例如，图1中的网络共有4台主机，则主机(2,1)对应的位置标识为(0,0,1,0)。通过将状态空间分解为单主机的状态信息表示，在一个有M台主机N个可执行操作的环境中，可以将状态空间大小从$O(3^{M\times N})$减少到$O(M\times 3^N)$。

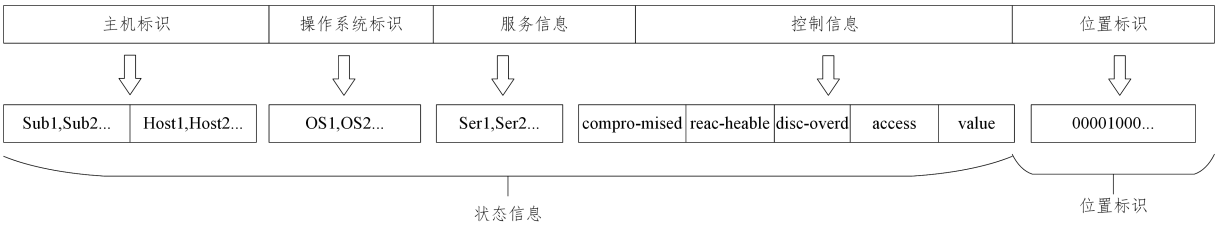


图4 状态空间的向量化表示

Fig. 4 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。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于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，可对状态空间中的操作系统字段进行更新，通过服务信息扫描，可确定主机运行过的服务，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。例如，表1中，对于图1中主机(1,0)的攻击动作包括4步，每个动作包含了动作类型、目标地址、动作成本、成功概率和成功执行后获取的主机权限。对于图1中的每个主机，攻击者可以根据表2所列的动作信息来生成针对不同主机的攻击动作。通过将攻击动作分解为上层主机选择和下层攻击动作选择，因此，对于具有M个目标主机和N个可执行操作的环境中的智能体，动作空间从原来的$O(M\times N)$减少到$O(M+N)$。

表 1 图 1 中攻击者的攻击动作示例

Table 1 Example of the attacker's attack action in Fig. 1

	攻击动作
$a_{1,1}$	OSScan: target=(1,0), cost=1.00, prob=1.00, req_access=None
$a_{1,2}$	ServiceScan: target=(1,0), cost=1.00, prob=1.00, req_access=None
$a_{1,3}$	Exploit: target=(1,0), cost=1.00, prob=0.80, req_access=1, os=linux, service=ssh, access=User
$a_{1,4}$	PrivilegeEscalation: target=(1,0), cost=1.00, prob=1.00, req_access=1, os=linux, process=tomcat, access=Root

表 2 图 1 中攻击者的动作列表

Table 2 Action list of the attacker in Fig. 1

Name	Type	Operation System	Cost	Probability	Access
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	—

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 3 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比较容易被攻击，漏洞利用所需的条件较少，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 等提出的方法[20]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。常见漏洞的利用成功概率示例如表 3 所列。

表 3 常见漏洞的利用成功概率示例

Table 3 Examples of successful exploitation probability of common vulnerabilities

漏洞	软件	攻击复杂度	概率设置
CVE-2021-35211	SSH	Low	0.8
CVE-2017-17405	FTP	Medium	0.5
CVE-2017-0145	SMB	High	0.2

下层奖励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，我们将奖励函数设置为智能体是否成功攻陷目标主机，如式(4)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励为 0；如果 s' 对应的状态为目标状态，则立即奖励设置为 1。即在规定步数内获取主机的 root 权限，将奖励值设置为 1，否则奖励值设置为 0。

$$r=\begin{cases}1, & \text{target state}\\ 0, & \text{other state}\end{cases} \tag{4}$$

通过给予下层内部奖励，激励下层策略学习到针对不同主机的攻击动作序列，从而可以有效缓解渗透测试过程中的稀疏奖励问题。

4.3 基于分层强化学习的网络攻击路径发现算法

4.3.1 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 4.2 节中的形式化表示。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励总和，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：

$$L_1(\theta_{1,i})=E_{(s_t,g_t,F_{t\sim t+N},s_{t+N},V_t)\sim D_1}[(y_{1,i}-Q_1(s,g;\theta_{1,i}))^2] \tag{5}$$

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

神经网络参数 $\theta_{1,i}$ 的更新公式为：

$$\nabla_{\theta_{1,i}} L_1(\theta_{1,i})=E_{(s,g,F,s',D)\sim D_1}[(f+\gamma \max_{g'} Q_1(s', g';\theta_{1,i-1})-Q_1(s,g;\theta_{1,i}))\nabla_{\theta_{1,i}} Q_1(s,g;\theta_{1,i})] \tag{6}$$

4.3.2 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励，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 $\theta_{2,j}$ 的更新式为：

$$L_2(\theta_{2,i})=E_{((s_t,g_t),a_t,r_t,(s_{t+1},g_t),v_t)\sim D_2}[(y_{2,i}-Q_2(s,a;\theta_{2,i},g))^2] \tag{7}$$

$$\nabla_{\theta_{2,i}} L_2(\theta_{2,i})=E_{((s,g),a,r,(s',g),v)\sim D_2}[(r+\gamma \max_{a'} Q_2(s', a';\theta_{2,i-1},g)-Q_2(s,a;\theta_{2,i},g))\nabla_{\theta_{2,i}} Q_2(s,a;\theta_{2,i},g)] \tag{8}$$

4.3.3 算法流程

基于分层强化学习的攻击路径发现算法如算法 1 所示。

算法 1 基于分层强化学习的攻击路径发现算法

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 θ_1，$Q_2(s,g,a;\theta_2)$ 参数 θ_2，上层 DQN 探索率 ϵ_1，下层 DQN 探索率 ϵ_2

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

1. for episode=1 to N do:

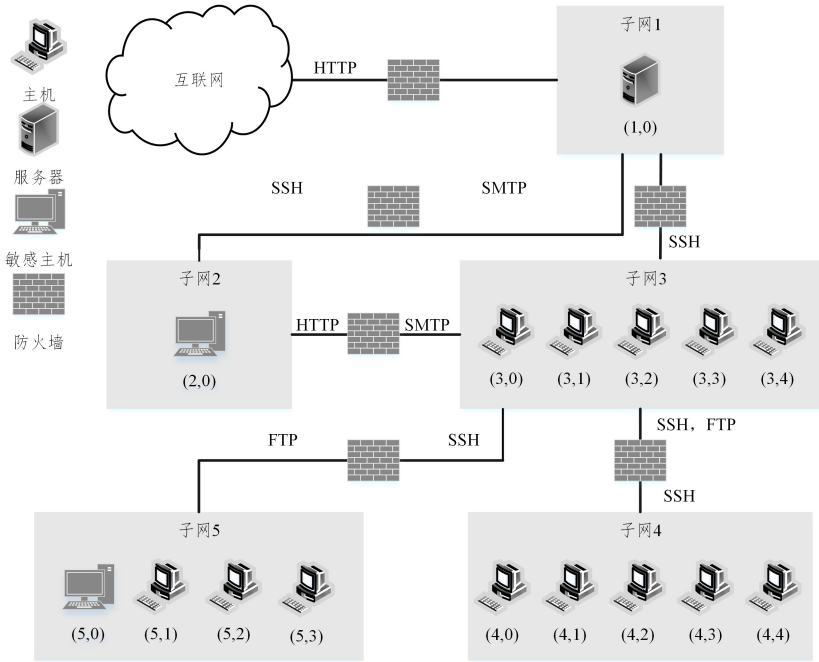
2. 初始化环境状态
3. for steps=1 to S do:
4. 选取目标主机g_t←ε-Greedy($Q(s_t,G;\theta_1)$,ε_1)
5. if 目标主机g_t没有被攻陷：
6. for goal_steps=1 to T：
7. 选取动作a_t←ε-Greedy($Q(s_t,g_t,A;\theta_2)$,ε_2)
8. 执行动作a_t，获取上层奖励 f 以及观察下一状态s_{t+1}
9. 获取下层奖励 r′←r(s,a,s′)
10. 存储序列({s,g},a,r′,{s′,g},v)到经验池D_2
11. 依据式(5)更新上层神经网络参数θ_1
12. 依据式(7)更新下层神经网络参数θ_2
13. F←F+f
14. end for
15. 存储序列(s,g,F,s′,V)到经验池D_1
16. elif 目标主机g_t已经被攻陷：
17. F=−10000
18. V=False
19. 存储序列(s,g,F,s′,V)到经验池D_1
20. 依据式(5)更新上层神经网络参数θ_1
21. end for
22. end for

5 实验分析

为了验证 iPathD 在不同网络场景下的性能，我们使用网络攻击模拟器 NASim[11] 作为测试平台。本文实验分为以下两部分进行：1)使用标准网络场景作为基准环境测试不同算法的性能；2)研究不同规模场景下算法的可扩展性。

5.1 实验设置

NASim 是一个开源的用于在网络安全环境中进行 AI 研究的实验平台，它旨在通过生成一系列模拟和抽象的网络场景来对自动化渗透测试代理进行测试。实验网络拓扑场景 1 模拟真实世界中的典型企业网络，共设置 2 个敏感主机作为渗透测试的目标，具体的网络拓扑如图 5 所示。



注：敏感主机为(2,0)和(5,0)。

图 5 实验场景 1

Fig. 5 Experimental scenario 1

实验场景 1 由 DMZ 区和内网两部分组成，其中子网 1 是 DMZ 区，其他子网为内网，共包含 5 个子网，16 台主机，敏感主机为(2,0)和(5,0)，即渗透测试的目标主机。我们通过配置防火墙规则来限制相邻子网之间的通信，仅允许部分特定的服务流量通过。

主机的配置信息如表 4 所列，选择在主机中运行的 5 个易受攻击的服务、3 个进程和 2 个操作系统来配置主机并定义它们的漏洞点，包括每台主机的虚拟地址(子网号和主机号)、操作系统类型和主机价值(衡量主机的重要性)，以及上面运行的各种服务和进程(用于攻击和提权的软件)。

为了研究算法在不同网络规模场景下的性能，我们分别在场景 1,2 不同主机规模的环境下进行了实验。敏感主机的数量固定为 2，因此随着网络规模的增大，奖励变得越来越稀疏。对于每台主机的攻击动作数为 Services+Processes+4，其中 4 表示子网、操作系统、服务和进程 4 种扫描操作。

表 4 场景 1 中每台主机的配置

Table 4 Configuration of each host inscenario 1

Address	Operation System	Host-value	Service	Process
(1,0)	Linux	0	HTTP	—
(2,0)	Windows	100	SMTP	Schtask
(3,0),(3,3),(3,4)	Windows	0	FTP	Schtask
(3,1)	Windows	0	FTP,HTTP	Daclsvc
(3,2)	Windows	0	FTP	—
(4,0),(4,1),(4,2),(5,2)	Linux	0	SSH	—
(4,3),(4,4)	Windows	0	SSH,FTP	Tomcat
(5,0)	Windows	100	SSH,SAMBA	Tomcat
(5,1)	Linux	0	SSH,HTTP	Tomcat
(5,3)	Linux	0	SSH	Daclsvc

当网络规模增大时，智能体除了受稀疏奖励问题的影响，还会受大规模动作空间问题的影响，而动作空间的大小除了受网络环境中主机的数量影响，还受漏洞的数量影响。因此除了比较在相同网络规模下的收敛效果，在场景 3 中，我们通过改变漏洞数量来比较算法的优越性。在场景 4 中，同时增加主机数量和漏洞数量，以比较算法的收敛性能。最后为了检验算法的可扩展性，我们在场景 5，6，7 中通过增加子网个数和主机个数，使用本文算法进行攻击路径发现，各场景的详细信息如表 5 所列。

表 5 实验网络场景列表

Table 5 Experimental network scenarios

Scenario	Hosts	Sensitive	Subnets	Services	Processes
Scenario 1	16	2	5	5	3
Scenario 2	35	2	10	5	3
Scenario 3	16	2	5	80	5
Scenario 4	35	2	10	80	5
Scenario 5	50	2	10	5	3
Scenario 6	100	2	15	5	3
Scenario 7	150	2	20	5	3

(1)在初始状态下，代理位于 Internet 上，不知道目标网络的全局信息。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。代理通过不断地试错来获得最大的累计奖励值，从而在每回合中有限的训练步骤内学习最佳攻击路径，每个 Episode 结束的条件是：1)获得所有敏感主机的 Root 权限；2)训练步数达到设定的最大值。

5.2 实验参数设置

实验环境是基于 Python3.7 开发的，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 6 所列。

表 6 超参数设置

Table 6 Hyperparameter settings

Hyperparameter	Meaning	Value
Learning rate/lr	学习率	0.00025
Batch size	训练选取的样本批次大小	256
Discount factor/γ	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	10000

5.3 实验结果分析

DQN 算法已经被广泛应用到现有的智能化渗透测试路径发现研究中[13,21]，文献[10]中基于 DQN 的改进算法较大程度地提高了攻击路径的发现效率。本实验以 DQN 算法和文献[10]提出的 DNSPI-DQN 算法为基准算法，对比分析 DQN 算法、DNSPI-DQN 算法以及 iPathD 算法在实验场景 1—4 下的模型复杂度和收敛性能。模型复杂度通过对比在不同场景中状态空间与动作空间的大小来进行分析；收敛性能的评价指标根据算法每回合的平均收益来分析，智能体在规定训练回合数上每回合的累计奖励值收敛越快，则算法效率越高。下面通过数据来对算法进行比较分析。

(1)模型复杂度分析

表 7 列出了分层 MDP 模型与单 MDP 模型在不同场景中状态空间与动作空间的对比情况。从表中可以明显看出，分层 MDP 模型明显优于单 MDP 模型，特别是随着网络中主机和漏洞数量的增多，分层 MDP 模型在求解效率上的优势越明显。

表 7 不同模型状态与动作空间大小对比

Table 7 Comparison of different model states and action spaces

实验场景	单 MDP 模型		分层 MDP 模型	
	状态空间	动作空间	状态空间	动作空间
Scenario 1	$3^{16\times8}$	192	16×3^{8}	28
Scenario 2	$3^{35\times8}$	420	35×3^{8}	47
Scenario 3	$3^{16\times85}$	1424	16×3^{85}	105
Scenario 4	$3^{35\times85}$	3115	35×3^{85}	124

(2)算法收敛性能对比分析

图 6 给出了每回合累积奖励值随训练回合数的变化。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，从而获取奖励值，因此奖励值可以用来衡量智能体的策略水平。从图 6 可以发现，在训练的初始阶段，智能体每回合所能获得的奖励值较小，但是随着训练的进行，奖励值不断增加，说明智能体逐渐学会了获取最大奖励值的策略。并且在前期探索过程中，iPathD 算法的每回合累计奖励值明显低于 DNSPI-DQN 和 DQN 算法，这是因为在 iPathD 算法中，上层策略的主机选择决定了下层策略能否攻击成功，在前期探索过程中，上层策略随机选择主机，当选择了无法攻击成功的主机后，下层策略必然攻击失败，也就浪费了大量的攻击步数，但是随着训练的进行，上层网络因为状态空间的大幅减小使得能快速学习到主机间的渗透过程。最终，iPathD，DQN 和 NDSPI-DQN 算法均在 1 000 回合以内收敛到了最大值累积奖励值，并且 iPathD 算法的性能最优，可以在 200 回合以内收敛到最优值。

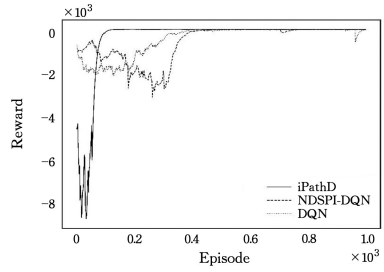


图 6 每回合累计奖励值随训练回合数的变化(场景 1)

Fig. 6 Cumulative reward value of each round varies with the number of training rounds(scenario 1)

图 7 给出了场景 1 中智能体最终选择的主机次数，目标“0”“1”“3”“12”分别表示主机(1,0)，(2,0)，(3,1)，(5,0)，智能体选择这些主机的次数远远多于其他主机，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(1,0)的 HTTP 漏洞获取权限，继而以(1,0)主机作为跳板，对子网 2 和子网 3 进行扫描，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，利用 FTP

漏洞攻击主机(3,1)，继而以主机(3,1)为跳板，扫描和攻击子网 5 中的敏感主机(5,0)。至此智能体只需攻击 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。

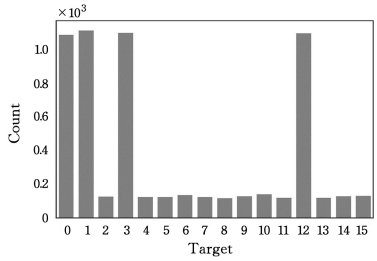


图 7 场景 1 中智能体目标主机的选择结果

Fig. 7 Agent target host selection result in scenario 1

图 8 给出了实验场景 2 的训练结果，在网络规模扩展后，DQN 和 NDSPI-DQN 的学习过程更加漫长，iPathD 能够在更少的训练回合中达到收敛。

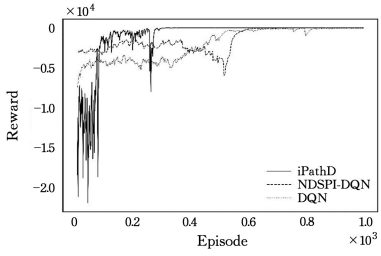


图 8 每回合累计奖励值随训练回合数的变化(场景 2)

Fig. 8 Cumulative reward value of each round varies with the number of training round(scenario 2)

图 9 给出了场景 3 的运行结果，可以看出 iPathD 和 NDSPI-DQN 明显优于 DQN 算法，并且 iPathD 的收敛速度更快。与场景 1 相比，在相同主机的情况下，即使增加主机的漏洞，对 iPathD 的收敛速度没有影响，因为 iPathD 的上层策略是根据主机的受控情况来选择下一步攻击目标的，屏蔽了下层单主机的攻击过程，即使下层主机的漏洞数量发生变化，对于上层策略来说没有影响，因此智能体的收敛速度不会发生太大变化，这充分体现了 iPathD 算法的优越性。

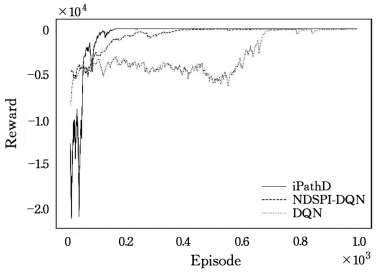


图 9 每回合累计奖励值随训练回合数的变化(场景 3)

Fig. 9 Cumulative reward value of each round varies with the number of training round(scenario 3)

图 10 给出了场景 4 的运行结果。从结果可以看出，除 DQN 算法以外，iPathD 和 NDSPI-DQN 均能在 1 000 回合内收敛到最大累计奖励值，并且 iPathD 算法的性能更优，可以在 400 回合以内收敛到最优值。

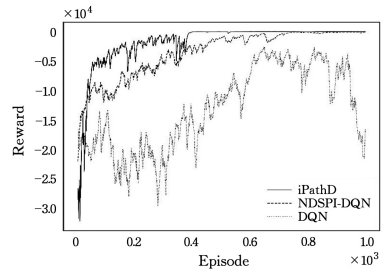


图 10 每回合累计奖励值随训练回合数的变化(场景 4)

Fig. 10 Cumulative reward value of each round varies with the number of training round(scenario 4)

(3)算法可扩展性结果分析

图 11 给出了使用 iPathD 算法训练的智能体在不同主机数量的网络场景下，每回合获得的平均累积奖励值随训练回合数的变化。从图中可以看到，iPathD 算法有较好的鲁棒性，在 3 个场景下其平均累积奖励值均能在 600 回合内收敛到最优值。但是随着子网个数和主机数的增加，因为智能体在每个回合需要尝试的主机数和动作数迅速增加，所以算法的收敛速度也在不断加快。在 150 台主机的模拟网络环境中，本文算法均能收敛到最优解。

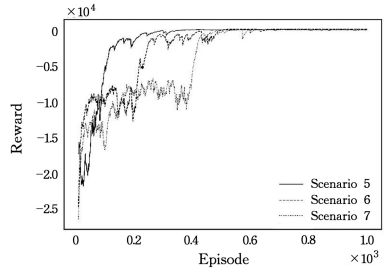


图 11 不同主机数量场景下平均累积奖励值随训练回合数的变化

Fig. 11 Average cumulative reward value varies with the number of training round under different host numbers

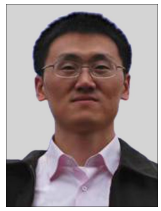
结束语 分层强化学习能够将复杂环境下的策略问题进行分解，逐步降低智能体的训练难度，成为了当前解决强化学习维度灾难和稀疏奖励的重要学习策略。通过将分层强化学习算法应用于渗透测试场景，有利于增强智能化攻击路径的发现效率，为自动化渗透测试提供一种新的研究方向。本文以此为研究背景，提出了一种基于分层 MDP 模型的 iPathD 智能化攻击路径发现算法，将渗透测试过程分解为主机间横向移动过程和单主机渗透过程，以获取主机权限为子目标，有效缓解了智能体状态空间爆炸、大规模动作空间和稀疏奖励的问题。实验结果表明，与传统 DQN 算法及其改进算法相比，iPathD 算法的收敛性能更好，同时具有一定的可扩展性，适用于较大规模的网络场景。当前基于强化学习的智能化渗透测试仍处于模拟验证阶段，无法证实其在真实环境下的有效性，并且没有考虑到攻防对抗条件下对攻击成功率的影响，因此实现在真实环境中以及考虑对抗条件下的智能化渗透测试是未来的研究方向。

参 考 文 献

[1] ARCE I, MCGRAW G. Guest editors' introduction: Why atta-

cking 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] SUTTON R S, BARTO A G. Reinforcement learling: An introduction[M]. MIT press, 2018.
[4] SARRAUTE C, BUFFET O, HOFFMANN J. Penetration testing==POMDP solving? [J]. arXiv: 1306.4714, 2013.
[5] SHMARYAHU D, SHANI G, HOFFMANN J, et al. Partially observable contingent planning for penetration testing[C]// Iwaise: First International Workshop on Artificial Intelligence in Security. 2017.
[6] SARRAUTE C, BUFFET O, HOFFMANN J. POMDPs make better hackers: Accounting for uncertainty in penetration testing [C]// Twenty-Sixth AAAI Conference on Artificial Intelligence. 2012.
[7] ZENNARO F M, ERDODI L. Modeling penetration testing with reinforcement learning using capture-the-flag challenges and tabular Q-learning[J]. arXiv: 2005.12632, 2020.
[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] HU Z, BEURAN R, TAN Y. Automated Penetration Testing Using Deep Reinforcement Learning[C]// IEEE European Symposium on Security and Privacy Workshops. 2020.
[10] ZHOU S, LIU J, HU D., et al. Autonomous Penetration Testing Based on Improved Deep Q-Network[J]. Appl. Sci. 2021, 11, 8823.
[11] SCHWARTZ J, KURNIAWATTI H. NASim: Network Attack Simulator[Z/OL]. https://networkattacksimulator.readthedocs.io/. 2019.
[12] SEIFERT C, BSTSER M, BLUM W, et al. CyberBattleSim[Z/OL]. https://github.com/microsoft/ cyberbattlesim, 2021.
[13] SCHWARTZ J, KURNIAWATI H. Autonomous penetration testing using reinforcement learning[J]. arXiv: 1905.05965, 2019.
[14] BARTO A G, MAHADEVAN S. Recent advances in hierarchical reinforcement learning[J]. Discrete Event Dynamic Systems, 2003, 13(1/2): 341-379.
[15] DAYAN P, HINTON G. Feudal Reinforcement Learning[C]// Proceedings of Advances in Neural Information Processing Systems. San Francisco: Morgan Kaufmann, 1993: 271-278.
[16] SINGH S. Transfer of Learning by Composing Solutions of Elemental Sequential Tasks[J]. Machine Learning, 1992, 8: 323-339.
[17] TAKAHASHI Y, ASADA M. Multi-controller Fusion in Multi-layered Reinforcement Learning[C]// International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI2001). Baden Baden, Germany, 2001: 7-12.
[18] CHEN T, LU J. Towards analysis of semi-Markov decision processes[C]// Artificial Intelligence and Computational Intelligence(AICI 2010). Berlin, Heidelberg: Springer, 2010: 41-48.
[19] MAHADEVAN S, MARCHALLECK N, DAS T, et al. Slef-improving Factory Simulation Using Continuous-time Average-reward Reinforcement Learning[C]// Proceedings of the 14th Internatioanl Conference on Machine Learning. Nashville, Tennessee, USA, 1997: 202-210.
[20] BACKES M, HOFFMANN J, KÜNNEMANN R, et al. Simulated penetration testing and mitigation analysis[J]. arXiv: 1705.05088.
[21] CHOWDHARY A, HUANG D, MAHENDRAN J S, et al. Autonomous security analysis and penetration testing[C]// 2020 16th International Conference on Mobility, Sensing and Networking(MSN). 2020: 508-515.

ZENG Qingwei, born in 1995, postgraduate. His main research interest is cyberspace security.



ZHANG Guomin, born in 1979, Ph.D, professor, master supervisor. His main research interests include software-defined networking, network security, network measurement, and distributed systems.

(责任编辑:喻藜)