基于深度强化学习 Noisy Net-A3C 算法的自动化渗透测试方法

董卫宇[1], 刘鹏坤[2], 刘春玲[1], 唐永鹤[1], 马钰普[2]

(1. 信息工程大学 网络空间安全学院,河南 郑州 450001;2. 郑州大学 网络空间安全学院,河南 郑州 450001)

摘　要:在自动化渗透测试领域,现有攻击路径决策算法大多基于部分可观测马尔可夫决策过程(POMDP),存在算法复杂度过高、收敛速度慢、易陷入局部最优解等问题。针对此问题,提出了一种基于马尔可夫决策过程(MDP)的强化学习算法 NoisyNet-A3C,并用于自动化渗透测试领域。该算法通过多线程训练 actor-critic,每个线程的运算结果反馈到主神经网络中,同时从主神经网络中获取最新的参数更新,充分利用计算机性能,减少数据相关性,提高训练效率。另外,训练网络添加噪声参数与权重网络训练更新参数,增加了行为策略的随机性,利于更快探索有效路径,减少了数据扰动的影响,增强了算法的鲁棒性。实验结果表明:与 A3C、Q-learning、DQN 和 NDSPI-DQN 算法相比,NoisyNet-A3C 算法收敛速度提高了 30%以上,验证了本研究算法的收敛速度更快。

关键词:渗透测试; 攻击路径决策; A3C 算法; 深度强化学习; Metasploit

中图分类号:TP181　　**文献标志码**:A　　**doi**:10.13705/j.issn.1671-6833.2024.02.011

渗透测试是一种针对信息系统实施的授权模拟攻击,旨在寻找威胁其安全的脆弱点,以便对其安全性进行评估。传统渗透测试过程过于依赖渗透测试人员的经验和操作,成本较高,而自动化渗透测试技术的出现在一定程度上缓解了传统渗透测试的不足。

强化学习(reinforcement learning, RL)是机器学习的一个分支,其灵感来源于心理学中的行为主义理论[1],涵盖了概率论、统计学、逼近论、凸分析、计算复杂性理论、运筹学等多领域的知识,是与人类学习最相似的一种学习形式[2]。Kröse 等[3]设计的 Q-learning 算法是诸多强化学习算法中的一种,能够在处理 MDP 问题时找到最优的动作选择策略,其 Q 值更新不受动作选择策略的影响,与通过值函数间接求解最优策略相比,策略梯度方法可以直接对收益函数求导来获得最优的梯度方向。Srinivasan 等[4]提出了 actor-critic 算法,通过引入优势函数,进一步提高了训练的稳定性。Kang 等[5]在 actor-critic 算法的基础上引入了多线程,提出了 A3CC 算法,多个线程可以同时学习最优策略,从而加快了训练的收敛速度。

近年来,强化学习算法被广泛应用于自动化渗透测试领域,主要用于探索自动化渗透测试的攻击路径。文献[6-10]设计了一种使用强化学习的自动化渗透测试方法,对网络进行渗透测试,可在一定程度上节省人力资源。Schwartz 等[11]提出了一种基于部分可观测马尔可夫决策过程(partially observable Markov decision process, POMDP)的自动化渗透测试框架,主要解决渗透测试过程中网络拓扑变化引发的问题。Mnih 等[12]对 Q-learning 算法进行改进,提出了 Deep Q-Network 算法。Zhou 等[13]借鉴 POMDP 理论框架,将渗透测试过程视为一个动态决策问题提出了一种改进的 DQN 算法,名为 NDSPI-DQN,用于解决大规模场景中的稀疏奖励问题,提高了收敛速度。Nguyen 等[14]提出了一种双代理架构方法,适用于多主机场景。

基于 POMDP 的自动化渗透测试方法虽然能够有效地反映渗透过程中的不确定性,更符合渗透测试这一过程,但具有求解复杂度高、收敛速度慢、对计算硬件的要求高、不适用于大规模网络等缺点。考虑到基于 MDP 的深度强化学习方法是对 POM-

收稿日期:2023-10-15;**修订日期**:2023-12-07

基金项目:国家重点研发计划项目(2018YFB080XXX);河南省重点研发项目(221111210300)

通信作者:刘春玲(1981—),女,河南滑县人,信息工程大学讲师,博士,主要从事漏洞挖掘与应用方向的研究,E-mail:lcl_509@163.com。

DP 的一种折中方法,能够兼顾渗透测试的成功率和求解效率,因此,本文尝试将基于 MDP 过程的 A3C 深度强化学习方法引入到自动化渗透测试场景中来,并对 A3C 算法进行改进,提出一种改进的强化学习算法 NoisyNet-A3C(Noisy Networks Integrated with Asynchronous Advantage Actor-Critic)。在 NoisyNet-A3C 算法模型中,智能体基于训练后的概率值选择最优的攻击路径,根据 Nmap 端口扫描信息确定漏洞和载荷,实时更新网络信息,不需要提前获取网络拓扑,从而使效率更高,渗透测试更为便捷。本文主要贡献如下。

(1)提出了改进的强化学习 NoisyNet-A3C 算法,通过建立多个智能体与环境交互,并最终更新到全局环境上,使用多线程执行多个智能体,减弱了训练智能体之间相关性过强的影响,进一步加快了训练过程中模型收敛速度。同时,通过引入 NoisyNet 机制,增加了模型探索能力,提高了训练收敛速度。

(2)设计并实现了一个自动化渗透测试框架,该框架包括扫描组件、模型训练与测试组件、漏洞探测与利用组件、数据库组件及 Metasploit 组件。NoisyNet-A3C 算法对应该框架的模型训练与测试组件。

1 NoisyNet-A3C 算法

MDP 和 POMDP 模型是一种通用框架,用于对渗透测试的不确定性进行建模分析。MDP 可以看作是 POMDP 的简化版本,其与 POMDP 相比在计算上更易于求解,这种计算效率的提升使得 MDP 在实践中应用更广泛[15]。寻找最佳攻击路径的方法是强化学习,能够通过与环境交互生成样本来优化性能[16]。因此将 MDP 与 RL 结合催生了一系列算法,其中就包括 Q-learning 算法、A3C 算法以及本文提出的对 A3C 算法改进的 NoisyNet-A3C 算法。

1.1 A3C 算法

A3C 算法是 Google DeepMind 团队提出的一种解决 actor-critic 算法不收敛问题的算法,是对 actor-critic 算法的进一步优化。在 actor-critic 算法中进行 2 组近似运算,第一组是将策略函数的近似为

$$\pi_{\theta(s,a)} = P(\boldsymbol{a} \mid \boldsymbol{s},\theta) \approx \pi(\boldsymbol{a} \mid \boldsymbol{s})。 \quad (1)$$

式中:策略 π 是一个包含参数 θ 的函数;s 表示某一状态;a 表示某一动作。

另一组是对价值函数的近似,对于动作价值函数和状态价值可近似为

$$\hat{v}(\boldsymbol{s},w) \approx v_{\pi(s)}; \quad (2)$$

$$\hat{q}(\boldsymbol{s},\boldsymbol{a},w) \approx q_{\pi(s,a)}。 \quad (3)$$

此时,策略函数被表示为一个连续的函数,对于连续函数,可以采用梯度上升法、牛顿法、拟牛顿法等优化方法寻找最优策略。最优化的目标就是使状态的收获的期望最大,这个状态可以是初始状态,可以是所有状态的平均价值,也可以是所有状态的平均奖励,无论采用哪种方法作为优化目标,最终对 θ 求导的梯度都可以表示为

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(\boldsymbol{s},\boldsymbol{a}) Q_\pi(\boldsymbol{s},\boldsymbol{a})]。 \quad (4)$$

在梯度更新部分 $\nabla_\theta \log \pi_\theta(\boldsymbol{s}_t,\boldsymbol{a}_t) Q_\pi$ 为分值函数,无需改变,$Q_\pi(\boldsymbol{s}_t,\boldsymbol{a}_t)$ 部分应从 critic 得到[17]。总的来说,actor 利用 v_t 更新策略函数的参数,基于概率选择动作,并得到奖励和新的状态,critic 基于 actor 的动作评判动作的奖励更新网络参数,actor 根据 critic 的奖励更新策略。

A3C 算法既可以处理连续和离散动作的问题,又可以单步更新,提升了学习效率。相较于 actor-critic 算法,A3C 主要基于 3 点进行优化:一是采用异步训练框架;二是优化了网络结构;三是对 critic 的优化。

异步训练框架如图 1 所示。A3C 将 actor-critic 放到多个线程中同步训练,充分利用并行的环境。将每个线程中的运行结果反馈给主网络,同时从主网络获取最新的参数更新,这样将多个线程结合在一起,同时进一步降低数据之间的相关性,加快训练速度,利于程序的收敛。

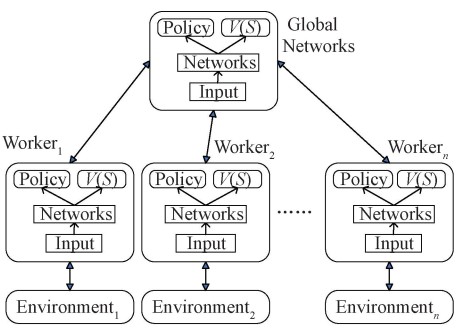

图 1 A3C 异步训练框架

Figure 1 A3C asynchronous training framework

对于网络结构的优化,与 actor-critic 算法不同的是,将 2 个网络放在一起,如图 2 所示。

图 2 A3C 网络结构优化

Figure 2 A3C network architecture optimization

对于 critic 评估点的优化,引入优势函数 A 作为

critic 评估点,优势函数在 A3C 算法中的表达式为

$$A(\boldsymbol{S},t)=\boldsymbol{R}_t+\boldsymbol{\gamma}\boldsymbol{R}_{t+1}+\cdots+\gamma^{n-1}\boldsymbol{R}_{t+n-1}+\gamma^n V(\boldsymbol{s}')-V(\boldsymbol{S})。\tag{5}$$

式中:t 表示时间;γ 为衰减因子;$\boldsymbol{R}$ 为奖励值;$V(\boldsymbol{s})$ 通过 critic 网络学习得到。

对于损失函数的优化主要是加入策略 π 的熵项,系数为 c,最终策略参数的梯度变为

$$\theta=\theta+\alpha\nabla_\theta\log\pi_\theta(\boldsymbol{s}_t,\boldsymbol{a}_t)A(\boldsymbol{S},t)+c\nabla_\theta H(\pi(\boldsymbol{S}_t,\theta))。\tag{6}$$

最后对 A3C 算法进行总结,其算法流程如下。

算法 1 A3C 算法。

输入:公共部分的 A3C 神经网络结构,对应参数位 θ、w,本线程的 A3C 神经网络结构,对应参数 θ'、w',全局共享的迭代轮数 T,全局最大迭代次数 $T_{\max}$,单次迭代时间序列最大长度 T_{local},状态特征 n,动作集 A,步长 α、β,熵系数 c,衰减因子 γ;

输出:策略 π 和价值函数 V。

① 更新时间序列 $t=1$

② 重置 Actor 和 Critic 的梯度更新量 $\mathrm{d}\theta\leftarrow 0,\mathrm{d}w\leftarrow 0$

③ 从公共部分的 A3C 神经网络同步参数到本线程的神经网络 $\theta'=\theta,w'=w$

④ $t_{\text{start}}=t$,初始化状态 s_t

⑤ 基于策略 $\pi(\boldsymbol{a}_t\mid\boldsymbol{s}_t,\theta)$ 选择出动作 $\boldsymbol{a}_t$

⑥ 执行动作 $\boldsymbol{a}_t$ 得到奖励 $\boldsymbol{r}_t$ 和新状态 $\boldsymbol{s}_{t+1}$

⑦ $t\leftarrow t+1,T\leftarrow T+1$

⑧ 如果 s_t 是终止状态,或 $t-t_{\text{start}}==t_{\text{local}}$ 则进入步骤 i,否则回退到步骤⑤

⑨ 计算最后一个时间序列位置 s_t 的 $Q(\boldsymbol{s},t)$:

$$Q(\boldsymbol{s},t)=\begin{cases}0, & terminal\,state\\ V(\boldsymbol{s}_t,w'), & none\,terminal\,state\end{cases}$$

⑩ for$i\in(t-1,t-2,\cdots t_s tart)$:

⑪ 计算每个时刻 $Q(\boldsymbol{s},i)$:$Q(\boldsymbol{s},i)=\boldsymbol{r}_i+\gamma Q(\boldsymbol{s},i+1)$

⑫ 累计 Actor 的本地梯度更新:

$$\mathrm{d}\theta\leftarrow\mathrm{d}\theta+\nabla_{\theta'}\log\pi_{\theta'}(\boldsymbol{s}_i,\boldsymbol{a}_i)(Q(\boldsymbol{s},i)-V(\boldsymbol{S}_i,w^i))+c\nabla_{\theta'}H(\pi(\boldsymbol{s}_i,\theta'))$$

⑬ 累计 Critic 的本地梯度更新:

$$\mathrm{d}w\leftarrow\mathrm{d}w+\frac{\partial(Q(\boldsymbol{s},i)-V(\boldsymbol{S}_i,w'))^2}{\partial w'}$$

⑭ 更新全局网络的模型参数:

$$\theta=\theta-\alpha\mathrm{d}\theta,w=w-\beta\mathrm{d}w$$

⑮ 如果 $T>T_{\max}$,结束循环输出公共部分的 A3C 神经网络参数 θ,w,否则进入步骤③。

1.2 NoisyNet-A3C 算法

A3C 算法的探索效率较低,训练过程过于盲目,容易陷入局部最优解,在自动化渗透测试领域效果并不显著。本文引入 NoisyNet 改进 A3C 算法,使之能更好地适应自动化渗透测试这一场景。

NoisyNet 是一种神经网络,其权值和偏差受到噪声参数的干扰,一般数学表示为 $y=f_\theta(x)$,是一个由噪声参数 θ 参数化的神经网络,本文将噪声参数 θ 表示为 $\theta=\mu+\sum\odot\boldsymbol{\varepsilon}$,其中 $\boldsymbol{\zeta}=(\mu,\sum)$ 定义为一组可学习的参数向量,$\boldsymbol{\varepsilon}$ 是一个具有固定统计量的零均值噪声向量,$\odot$ 表示逐元素乘法。神经网络的损失函数被重新表示为

$$\boldsymbol{\varepsilon}:\bar{L}(\boldsymbol{\zeta})=E[L(\theta)]。\tag{7}$$

本文对 $\boldsymbol{\zeta}$ 进行优化,带有噪声参数的全连接层数学表示如下:

$$y=(\boldsymbol{\mu}^w+\boldsymbol{\sigma}^{w\odot\boldsymbol{\varepsilon}^w})x+\boldsymbol{\mu}^b+\boldsymbol{\sigma}^b\odot\boldsymbol{\varepsilon}^b。\tag{8}$$

式中:$\boldsymbol{\mu}^w+\boldsymbol{\sigma}^{w\odot\boldsymbol{\varepsilon}^w}$ 等价于 w,$\boldsymbol{\mu}^b+\boldsymbol{\sigma}^b\odot\boldsymbol{\varepsilon}^b$ 等价于 b,$\boldsymbol{\mu}^w\in\mathbf{R}^{q\times p}$,$\boldsymbol{\sigma}^w\in\mathbf{R}^{q\times p}$,$\boldsymbol{\mu}^b\in\mathbf{R}^q$,$\boldsymbol{\sigma}^b\in\mathbf{R}^q$ 是可学习的,$\boldsymbol{\varepsilon}^w\in\mathbf{R}^{q\times p}$ 和 $\boldsymbol{\varepsilon}^b\in\mathbf{R}^q$ 为噪声随机变量。

上述主要陈述如何引入噪声,在强化学习中,智能体需要在探索未知环境和利用已知信息之间进行平衡[18]。独立高斯噪声可以用于增加行为策略的随机性,鼓励代理探索新的行动,而不仅仅依赖已知的最佳策略,这有助于避免代理过早陷入局部最优解,并提高了训练的稳定性和收敛速度。因此,本文主要引入独立高斯噪声。噪声层每个权重都是独立的,并且具有模型自己学习的 μ 和 σ,也就是对于任意的 $\boldsymbol{\varepsilon}^w_{i,j}$ 和 $\boldsymbol{\varepsilon}^b_j$ 的参数都来自高斯分布。

为了将 NoisyNet 引入 A3C 算法中,本文对 A3C 做了如下修改:首先,删除了策略损失的熵加成;其次,将策略网络的全连接层用独立高斯噪声参数化为噪声网络层,本文将这种模型称为 NoisyNet-A3C,具体改进如下。

A3C 算法中,为了训练策略,需要为每个状态计算策略梯度的近似值,策略梯度 $U(\theta)$ 如下:

$$U(\theta)=\nabla_{\theta_\pi}\log(\pi(\boldsymbol{a}_{t+i}\mid\boldsymbol{s}_{t+i};\theta_\pi))[\hat{Q}_i-V(x_{t+i};\theta_V)]。\tag{9}$$

式中:$\hat{Q}_i$ 是对 $\hat{Q}_i=\sum_{j=i}^{k}\gamma^{j-i}\boldsymbol{r}_{t+j}+\gamma^{k-i}V(x_{t+k};\theta_V)$ 的估计,然后将梯度相加,得到累积梯度:

$$\sum_{i=0}^{k}U(\theta)。\tag{10}$$

A3C 通过最小化估计回报与价值之间的误差来训练价值 $\sum_{i=0}^{k}(\hat{Q}_i-V(\boldsymbol{x}_{t+i};\theta_V))^2$,然后更新网络参数 (θ_π,θ_V),推出如下更新公式:

$$\theta_\pi \leftarrow \theta_\pi + \alpha_\pi \sum_{i=0}^{k} U(\theta)。 \tag{11}$$

$$\theta_V \leftarrow \theta_V - \alpha_V \sum_{i=0}^{k} \nabla_{\theta_V} (\hat{Q}_i - V(\boldsymbol{x}_{t+i};\theta_V))^2。 \tag{12}$$

在 NoisyNet-A3C 算法中,熵损定义为

$$\beta \sum_{i=0}^{k} \nabla_{\theta_\pi} H(\pi(. \mid \boldsymbol{x}_{t+i};\theta_\pi))。 \tag{13}$$

式中:熵损失中 $H(\pi(. \mid \boldsymbol{x}_{t+i};\theta_\pi)) = -\beta \sum_{a \in A} \pi(\boldsymbol{a} \mid \boldsymbol{s}_{t+i};\theta_\pi) \log(\pi(a \mid \boldsymbol{s}_{t+i};\theta_\pi))$。通过熵损失更新策略,熵损失能够提升强化学习过程中的探索效率,当采用带有噪声层替换价值和策略中的线性层时,效果更佳,最终对 A3C 算法中值函数优化为

$$\widehat{Q}_i = \sum_{j=i}^{k-1} \gamma^{j-i} \boldsymbol{r}_{t+j} + \gamma^{k-i} V(\boldsymbol{x}_{t+k};\boldsymbol{\zeta}_V,\boldsymbol{\varepsilon}_i)。 \tag{14}$$

$\widehat{Q}_i$ 是当前策略回报的一致估计值。为了达到这个目的,本文强制 $\forall i, \boldsymbol{\varepsilon}_i = \boldsymbol{\varepsilon}$。由于 A3C 算法是一个强化学习算法,这意味着需要固定噪声和策略。因此,参数 $\boldsymbol{\zeta}_\pi$,$\boldsymbol{\zeta}_V$ 的每次更新都是在整个网络的噪声保持不变的情况下进行的,以便策略保持不变,最终参数更新公式修改为

$$U(\boldsymbol{\zeta},\boldsymbol{\varepsilon}) = \nabla_{\zeta_\pi} \log(\pi(\boldsymbol{a}_{t+i} \mid \boldsymbol{s}_{t+i};\boldsymbol{\zeta}_\pi,\boldsymbol{\varepsilon})) \cdot [\hat{Q}_i - V(\boldsymbol{x}_{t+i};\boldsymbol{\zeta}_V,\boldsymbol{\varepsilon})]; \tag{15}$$

$$\boldsymbol{\zeta}_\pi \leftarrow \boldsymbol{\zeta}_\pi + \alpha_\pi \sum_{i=0}^{k} U(\boldsymbol{\zeta},\boldsymbol{\varepsilon}); \tag{16}$$

$$\boldsymbol{\zeta}_V \leftarrow \boldsymbol{\zeta}_V - \alpha_V \sum_{i=0}^{k} \nabla_{\zeta_V} (\hat{Q}_i - V(\boldsymbol{x}_{t+i};\boldsymbol{\zeta}_V,\boldsymbol{\varepsilon}))^2。 \tag{17}$$

NoisyNet-A3C 算法通过不断与环境进行交互获得奖励值,以最大化 $\widehat{Q}_i$ 为目标,根据奖励不断更新参数 $\boldsymbol{\zeta}_\pi$ 和 $\boldsymbol{\zeta}_V$,从而选出最优的攻击策略。

2 基于 NoisyNet-A3C 算法的自动化渗透测试框架

本节将介绍基于 NoisyNet-A3C 算法的自动化渗透测试框架。该框架包括 5 个组件(如图 3 所示):①扫描组件:使用 Nmap 扫描并解析网络端口信息,构建端口列表和操作系统列表,存储在数据库中;②Metasploit 组件:执行具体攻击操作;③数据库组件:存储渗透测试攻击路径的相关数据和数据表间的对应关系;④漏洞探测与利用组件:执行漏洞探测和攻击操作,如果攻击成功,进一步探测子网的 IP 地址;⑤模型训练与测试组件:主要用于模型的训练和测试。

2.1 扫描组件

使用深度强化学习的关键在于有训练和测试数据,本文通过 Nmap 扫描模块对目标设备进行扫描,

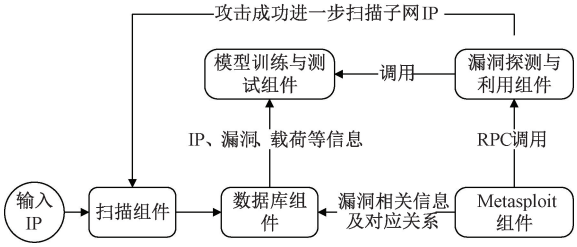

图 3 基于 NoisyNet-A3C 算法的自动化渗透测试框架

Figure 3 Penetration testing automation framework based on NoisyNet-A3C algorithm

并对 Nmap 扫描结果进行解析,以获取漏洞探测与利用组件所需的 IP 信息。扫描模块完成扫描和解析后,获得的开放端口列表和操作系统信息如下。

```
port_list:[
    {'portid': '21', 'protocol': 'tcp', 'version': 0.0, 'prod_name': 'ftp', 'product': ftpd'},
    ...
]
os_info:{
    'rhost': '192.168.79.129',
    'os_name': 'windows',
    'os_version': 'Microsoft Windows Server 2008 SP2 or Windows 10 or Xbox One...'
}
```

将解析后的数据存入数据库中,后续漏洞探测与利用组件根据这些信息执行攻击行为,模型训练与测试组件将这些信息作为模型输入。

2.2 Metasploit 组件

Metasploit 组件是一个能够查找、利用和验证漏洞的渗透测试平台,其本身附带数百个已知软件漏洞以及专业级漏洞攻击工具,被广泛用于漏洞探测与利用过程。数据库组件根据 Metasploit 组件获取已知的漏洞信息,漏洞探测与利用组件通过 RPC 调用 Metasploit 组件执行具体模拟攻击操作。

2.3 数据库组件

数据库设计主要分为 exploits 攻击模块表、targets 模块表、payloads 模块表和额外拓展表,主要存储 exploit、target、payload 在 Metasploit 组件中的索引以及各自对应关系信息,便于执行漏洞探测与利用操作。数据表设计以及相应对应关系如图 4 所示。

2.4 漏洞探测与利用组件

漏洞探测与利用组件依据 NoisyNet-A3C 模型训练与测试组件选择的 exploit、target、payload 信息顺序执行漏洞利用程序,攻击完成后,获取模块的攻击结果,判断是否成功获取到 session,获取成功则为

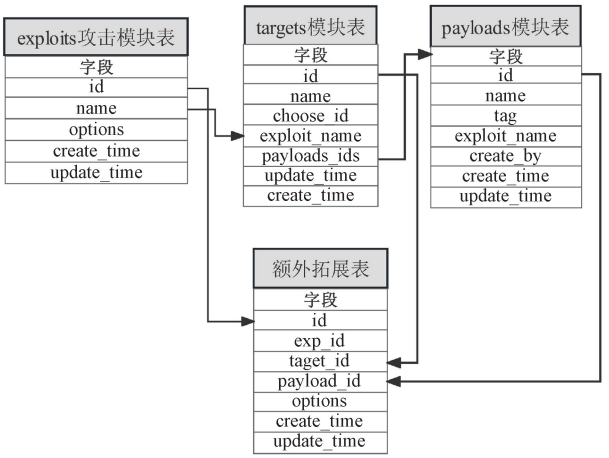

图 4　漏洞相关数据表设计

Figure 4　Design of the vulnerability-related data table

获取到的 session 配置代理,并将目标 IP 重新提交给扫描组件。

2.5　模型训练与测试组件

该组件从数据库中获取数据,将数据向量化、正则化后作为模型的输入数据。按照输入数据对应的 exploit 信息、target 信息、payload 信息执行漏洞探测与利用组件,实时获取漏洞探测和攻击行为的奖励,并以最大化奖励值为训练目标来训练 Noisy-Net-A3C 模型。

3　基于 NoisyNet-A3C 算法的自动化渗透测试框架的实现

本章节旨在详细阐述基于 NoisyNet-A3C 算法的自动化渗透测试框架的具体实现流程,从漏洞探测与利用组件的设计以及 NoisyNet-A3C 模型训练与测试组件的设计 2 个方面进行论述。

3.1　漏洞探测与利用组件设计

漏洞探测与利用组件是程序的主要组件之一,在程序启动后驻留在后台,等待扫描组件生成攻击表。在获取到攻击目标数据之后,该组件会对数据进行向量化和归一化处理。处理后的数据将经过 NoisyNet-A3C 算法的训练,根据渗透测试攻击路径攻击成功概率筛选排序,生成从优到劣的攻击方案。然后,按攻击方案对目标进行攻击,最终以获得 sessions。

在成功获取 sessions 后,将跳出该目标的攻击循环,通过这个 sessions 发送指定的命令,收集主机的基本信息,并利用代理模块配置攻击代理,将收集到的新的待攻击 IP 地址加入到扫描组件的待扫描列表,等待扫描组件处理后重复 NoisyNet-A3C 算法训练操作。目前,漏洞探测与利用组件配置代理进行跨网段攻击仅支持在经过 1 层代理的网络上。漏洞探测与利用组件的设计如图 5 所示。

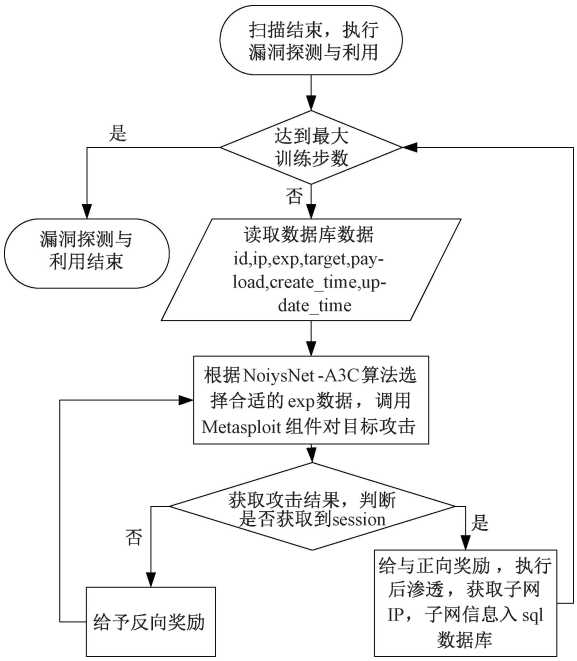

图 5　漏洞探测与利用组件设计

Figure 5　Design of vulnerability detection and exploitation components

3.2　模型训练与测试组件设计

在训练模式中,将“service_name”、“service_version”、“exploit_id”和“target_id”作为状态“state”,其中“service_name”和“service_version”是在扫描组件识别的设备的服务和版本;确定“service_name”后,通过语句“search name:+ service_name type:exploit app:server”在 Metasploit 中查询并随机选择一个“exploit”模块,确定“exploit_id”;根据 Nmap 扫描得到的主机类型,在可选的“target”列表中选择“target_id”;在确定了这些状态之后,NoisyNet-A3C 算法根据状态计算要执行的动作(payload)的概率,选择概率最高的 payload 并调用 Metasploit 执行漏洞利用,循环上述过程,通过不断修正神经网络参数来训练出最优的决策路径。整个模型训练流程详见图 6。

在测试模式下,根据 $\pi(s)$ 从模型中选择决策路径,提高攻击成功率。此外,测试模式还增加了后渗透阶段,计算不同状态空间下 payload 选择的概率,并根据概率从大到小调用 Metasploit 进行渗透测试,若渗透测试成功,则进行后渗透,识别出内网的存活的主机,并调用 Metasploit 框架中自带的代理模块“auxiliary/server/socks4a”搭建代理,对新识别到的主机进行进一步渗透测试,直到没有新的主机为止。

4　实验

为了验证 NoisyNet-A3C 作为攻击路径决策算

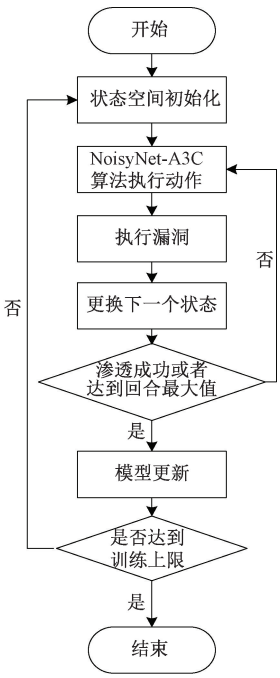

图 6　NoisyNet-A3C 模型训练流程图

Figure 6　Flowchart of NoisyNet-A3C model training process

法在渗透测试框架中的效果,本文在 EVE-NG 网络仿真平台上设计并实现了一种网络拓扑,并测试 NoisyNet-A3C、A3C、Q-learning、DQN 以及 2021 年 Zhou 等[13]提出的 NDSPI-DQN 算法在该网络拓扑中渗透测试的性能。

4.1　实验场景

本文尽可能地模拟真实内网环境,设计了一个由 12 台网络设备组成的网络拓扑环境。这些网络设备包括主机以及具有任意操作系统和应用程序的设备。图 7 展示了一个由 12 台设备(5 台计算机、4 台交换机和 3 台路由器)组成的网络拓扑,攻击者扫描网络拓扑的 IP 地址,以识别敏感设备,识别出设备漏洞,生成漏洞报告。

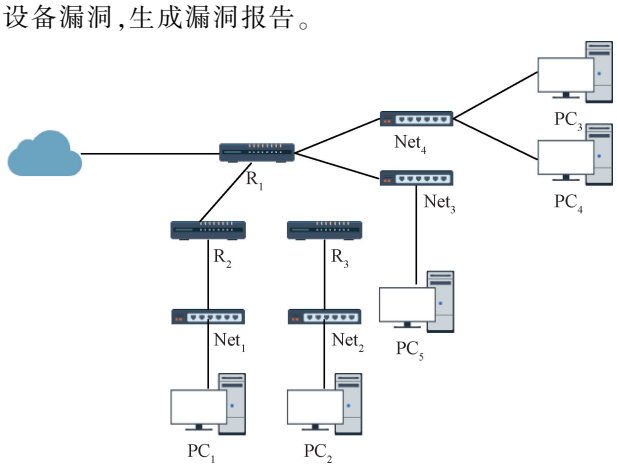

图 7　网络拓扑图

Figure 7　Network topology diagram

本文旨在评估所提出的基于 MDP 模型的自动化渗透测试框架的有效性和收敛速度。为此,需要在网络设备中植入不同的网络漏洞,并建立网络设备、设备操作系统与漏洞及其开放端口的对应关系,如表 1 所示。

表 1　设备漏洞对应关系

Table 1　Device vulnerability correlation

设备名	操作系统	端口	漏洞
PC_1	Windows 7	445	ms17-010
		80	ms15-034
PC_2	Windows xp	445	ms08-067
PC_3	Windows Server 2003 R2	1 039	CVE-2017-7269
PC_4	Metasploitable2(kali)	80	CVE-2014-6271
PC_5	Windows 10	445	CVE-2020-0796

本文针对上述漏洞进行自动化渗透测试,测试结果显示,该漏洞在训练和测试阶段均能正常渗透。本文提出的基于 NoisyNet-A3C 算法的自动化渗透测试框架具有显著的有效性。接下来,本文将评估算法训练的收敛速度。

4.2　实验设置

本节主要介绍实验的硬件配置和超参数的设定。超参数包括学习率、Episode、损失因子及学习次数。本文通过合适的超参数设定,使模型取得优良的效果。

实验采用的软件环境为 Ubuntu18.04 操作系统、python3.6.9、Metasploit Framework 6.1.34、tensorflow1.8.0、Keras2.1.6 和 pandas0.23.0。硬件环境为 Genuine Intel(R) CPU @ 2.00 GHz,32 GB 内存,Tesla K80 GPU 11 GB 显存。

(1)学习率的设置。本文在评估 NoisyNet-A3C 算法的收敛速度的基础上,确定了相应的学习率,以期能使模型的效果得到提升。经实验证明,将学习率设置为 0.005,能够使训练进程快速收敛且避免陷入局部最优解的境地。

(2)回合(episode)步数的设置。本文采用 NoisyNet-A3C 算法进行训练,该算法的训练以 episode 为基础。具体而言,以状态 state 到 episode 结束时的返回值作为目标值,拟合到 V 网络,并利用 V 网络对 policy 网络进行更新。为了保证训练的有效性,本文将 episode 的值设置为 2 000。

(3)损失因子的设置。NoisyNet-A3C 算法的损失函数主要指的是为熵损失。熵损失是一个非常重要的超参数,对最终性能和收敛速度有直接影响。通过实验结果显示,将熵损失设置为 0.01 时性能最佳,这不仅不会导致策略熵快速下降并陷入局部最

优解,从而无法学习到有效数据,也不会使策略熵下降过慢,从而使有效数据被随机性掩盖。

(4) 学习次数的设定。本文将总训练步数设为 80 000。通过实验验证,80 000 次训练已经足以证明算法的收敛性,并且不会过度消耗计算机资源。

(5)优化器的设定。优化器决定了神经网络的优化算法,包括 SGD、Adam、RMSprop 等。Adam 优化器是一种常用于深度学习模型训练的优化算法,它结合了 SGD 和 RMSprop 的优势,具有自适应学习率、对稀疏梯度的鲁棒性、并行性等优点。在 Noisy-Net-A3C 算法中,智能体通常只与环境交互一次,梯度通常是稀疏的,并且并行性具有显著提高训练效率的优势。因此,本文 NoisyNet-A3C 算法优化器设置为 Adam。

(6)折扣因子的设置。折扣因子通常用符号 γ 表示,在强化学习中用来调整近期和远期奖励的影响,其取值为(0,1]。本文将折扣因子设置为 0.955。通过实验证明,当折扣因子设置为 0.9 时,由于折扣因子过低,算法忽略了长期目标,从而使得收敛速度减缓。而当折扣因子设置为 0.99 时,折扣因子过高,使得算法陷入了局部最优解,在局部最优解附近不断徘徊。经过实验,折扣因子设置为 0.955 时,效果最佳。不同折扣因子在实验中的平均奖励情况详见图 8。

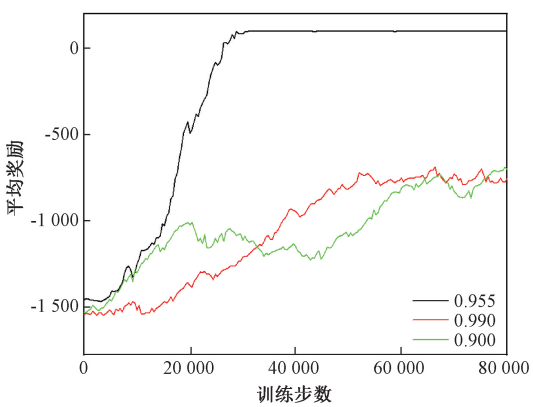

图 8　不同折扣因子下平均奖励变化

Figure 8　Variation average reward with different discount factors

4.3　实验结果

强化学习的目标是通过训练优化策略,使得模型从环境中获得的奖励的期望值达到最大化[19]。因此,本文选择在规定训练步数内平均奖励的变化作为评估指标。当模型在环境中获得了奖励,就会结束该回合,并给予正面的反馈。如果训练模型收敛,每回合的步数会逐渐减少。因此,本实验选择回合步数的变化作为另一个评估指标。

本文在构建的实验场景中,使用了相同的超参数对 NoisyNet-A3C、A3C、Q-learning、DQN 以及文献[13]中提到的 NDSPI-DQN 算法进行了评估。平均奖励值的大小是评价算法性能的重要因素。平均奖励值随训练步数的变化,参见图 9。

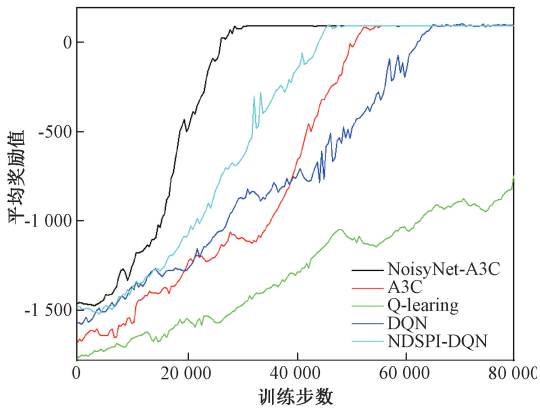

图 9　平均奖励随训练步数的变化

Figure 9　Variation of average reward with training steps

如图 9 所示,所有的模型在训练步数增加时,平均奖励值都有所升高。随着训练步数的增加,模型能够从环境中学习到更多的信息,从而优化其策略。Noisy-A3C 算法在训练中表现出比 A3C 更好的性能,平均奖励值提前大约 18 000 步收敛,这是因为其引入的噪声可以帮助智能体探索环境,从而更快地学习,可以在更复杂的环境中学习和优化策略。与 NDSPI-DQN 算法和 DQN 算法相比,Noisy-A3C 算法平均奖励值收敛提前 10 000 步。这主要因为 Noisy Net-A3C 不具有经验池的概念,更具有实时性,这意味着它可以更快地适应环境的变化,能够更及时地更新策略,从而更好地应对环境的变化。

实验选取的另一个评价指标是每回合训练步数的变化。当模型从环境中能有所得时,给予正反馈结束该回合,因此回合步数是逐渐减少的。回合步数的变化如图 10 所示。

在本文设计的自动化渗透测试框架中,NoisyNet-A3C 算法是对 A3C 算法的改进。因此,实验对 A3C 算法和 NoisyNet-A3C 算法进行了比较。实验结果表明,NoisyNet-A3C 算法的收敛速度更快,收敛更为稳定,相对于 A3C 算法,其提前收敛的训练步数超过 15 000 步,优化效果显著。本文在实验中证明了 NoisyNet-A3C、DQN 和 NDSPI-DQN 3 种算法均具有收敛至最优状态的能力。其中,NoisyNet-A3C 算法具有最快的收敛速度,与 DQN 算法相比,Noisy-Net-A3C 算法提前收敛了大约 23 000 步;与 NDSPI-DQN 算法相比,提前收敛了大约 8 000 步。然而,Q-

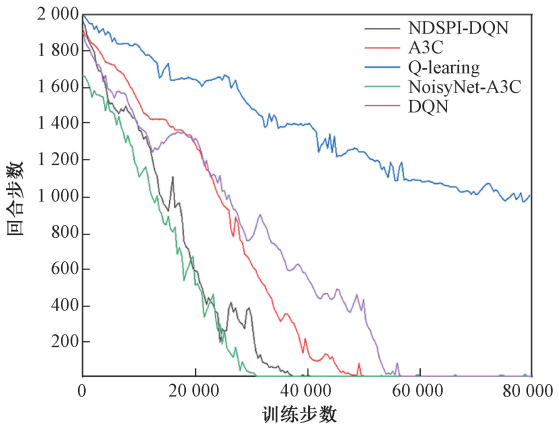

图 10　每回合步数随训练步数的变化

Figure 10　Variation of number of steps per round with training steps

learning 算法在实验中的训练步数限制下未能达到最优状态,可能需要增加训练步数。由于 Q-learning 算法仅作为对比算法,因此本实验没有对其进行过多的训练。相较于传统的 A3C 算法,NoisyNet-A3C 主要通过在神经网络权重上加入参数化的噪声,可以更快地适应环境的变化,并及时更新策略。相比之下,A3C 可能需要更多的训练步数才能逐渐适应环境并学习到有效的策略。与 Q-learning、DQN 和 NDSPI-DQN 相比,NoisyNet-A3C 没有经验池的概念,更具有实时性,结合了多线程以及基于策略和基于值的方法的优点,降低了数据之间的相关性,进一步加速了收敛速度。

3　结论

(1)本文提出了一种适用于自动化渗透测试场景的 NoisyNet-A3C 算法。对比 NoisyNet-A3C、A3C、Q-learning、DQN 和 NDSPI-DQN 5 种算法,本文提出的 NoisyNet-A3C 算法提前收敛的训练步数超过 8 000 步,收敛速度提高了 30%以上,收敛曲线波动更小。NoisyNet-A3C 算法具有两大优势:首先,它可以提供更高精度的预测、更快的训练速度、更好的泛化能力,从而更好地抵抗过拟合,同时还可以提供更好的健壮性和抗噪声能力,从而更好地处理不确定性和噪声;其次,它可以在多个环境中并行运行,提高训练速度,可以更有效地学习环境的复杂性,并且可以更快地收敛到最优策略。

(2)设计并实现了一个自动化渗透测试框架。在该框架中对 NoisyNet-A3C 算法的性能进行验证,此外,本框架所需的算法漏洞载荷信息来源于 Metasploit,攻击信息更为充分,能覆盖更多的设备和协议。

参考文献:

[1] 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.

[2] 黄万伟, 郑向雨, 张超钦,等. 基于深度强化学习的智能路由技术研究[J]. 郑州大学学报(工学版),2023, 44(1):44-51.
HUANG W W, ZHENG X Y, ZHANG C Q, et al. Research on intelligent routing technology based on deep reinforcement learning[J]. Journal of Zhengzhou university (engineering science),2023,44(1):44-51.

[3] KRÖSE B J A. Learning from delayed rewards[J]. Robotics and Autonomous Systems, 1995, 15(4): 233-235.

[4] SRINIVASAN S, LANCTOT M, ZAMBALDI V, et al. Actor-critic policy optimization in partially observable multiagent environments[C]//Proceedings of the 32nd International Conference on Neural Information Processing Systems. NewYork:ACM, 2018: 3426-3439.

[5] KANG Q M, ZHOU H Z, KANG Y F. An asynchronous advantage actor-critic reinforcement learning method for stock selection and portfolio management[C]//Proceedings of the 2nd International Conference on Big Data Research. NewYork:ACM, 2018: 141-145.

[6] GHANEM M C, CHEN T M. Reinforcement learning for intelligent penetration testing[C]//2018 Second World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4). Piscataway:IEEE, 2018: 185-192.

[7] 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.
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.

[8] KANG Q M, ZHOU H Z, KANG Y F. An asynchronous advantage actor-critic reinforcement learning method for stock selection and portfolio management[C]//Proceedings of the 2nd International Conference on Big Data Research. NewYork:ACM, 2018: 141-145.

[9] HU Z G, BEURAN R, TAN Y S. Automated penetration testing using deep reinforcement learning[C]//2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). Piscataway:IEEE, 2020: 2-10.

[10] CHEN J Y, HU S L, ZHENG H B, et al. GAIL-PT: a generic intelligent penetration testing framework with generative adversarial imitation learning[EB/OL]. (2022-

04 - 05)[2023 - 06 - 11] https://arxiv.org/abs/2204.01975.pdf.

[11] SCHWARTZ J, KURNIAWATI H, EL-MAHASSNI E. POMDP + information-decay: incorporating defender's behaviour in autonomous penetration testing[J]. Proceedings of the International Conference on Automated Planning and Scheduling, 2020, 30: 235-243.

[12] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.

[13] 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.

[14] NGUYEN H, TEERAKANOK S, INOMATA A, et al. The proposal of double agent architecture using actor-critic algorithm for penetration testing[C]//Proceedings of the 7th International Conference on Information Systems Security and Privacy. San Francisco: Science and Technology Publications, 2021: 440-449.

[15] JIANG Z Y, ZHANG T J, KIRK R, et al. Graph backup: data efficient backup exploiting Markovian transitions[EB/OL]. (2022-05-31)[2023-06-11]. https://arxiv.org/abs/2205.15824.pdf.

[16] 王丙琛, 司怀伟, 谭国真. 基于深度强化学习的自动驾驶车控制算法研究[J]. 郑州大学学报(工学版), 2020, 41(4): 41-45, 80.
WANG B C, SI H W, TAN G Z. Research on autopilot control algorithm based on deep reinforcement learning[J]. Journal of Zhengzhou University (Engineering Science), 2020, 41(4): 41-45, 80.

[17] SUTTON R S, MCALLESTER D, SINGH S, et al. Policy gradient methods for reinforcement learning with function approximation[C]//Proceedings of the 12th International Conference on Neural Information Processing Systems. NewYork: ACM, 1999: 1057-1063.

[18] YE D H, CHEN G B, ZHANG W, et al. Towards playing full MOBA games with deep reinforcement learning[C]//Proceedings of the 34th International Conference on Neural Information Processing Systems. NewYork: ACM, 2020: 621-632.

[19] 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(7587): 484-489.

Automated penetration testing method based on deep reinforcement learning Noisy Net-A3C algorithm

DONG Weiyu[1], LIU Pengkun[2], LIU Chunling[1], TANG Yonghe[1], MA Yupu[2]

(1. School of Network and Cybersecurity, Information Engineering University, Zhengzhou 450001, China; 2. School of Network and Cybersecurity, Zhengzhou University, Zhengzhou 450001, China)

Abstract: In the field of automated penetration testing, most existing attack path decision algorithms are based on partially observable Markov decision processes (POMDP), which have problems such as high algorithm complexity, slow convergence speed, and susceptibility to getting stuck in local optima. This article proposes a reinforcement learning algorithm NoisyNet-A3C based on Markov Decision Process (MDP) and applies it to the field of automated penetration testing. This algorithm trains actor-critic through multiple threads, and the operation results of each thread are fed back to the main neural network. At the same time, the latest parameter updates are obtained from the main neural network, fully utilizing computer performance, reducing data correlation, and improving training efficiency. In addition, adding noise parameters and weight network training update parameters to the training network increases the randomness of the behavior strategy, facilitates faster exploration of effective paths, reduces the impact of data disturbances, and enhances the robustness of the algorithm. The experimental results show that compared with A3C, Q-learning, DQN, and NDSPI-DQN algorithms, the NoisyNet-A3C algorithm converges more than 30% faster, verifying that the algorithm proposed in this paper converges faster.

Keywords: penetration testing; attack path decision; A3C algorithm; deep reinforcement learning; Metasploit