基于深度强化学习的微服务多维动态防御策略研究

周大成，陈鸿昶，何威振，程国振，扈红超
（信息工程大学信息技术研究所，河南 郑州 450002）

摘 要：针对云原生中安全防御策略在动态请求流量下难以兼顾服务质量的问题，提出基于深度强化学习的微服务多维动态防御策略，简称 D2RA 策略，在流量动态变化时给出兼顾安全防御和服务质量的动态配置方案。首先，基于微服务多副本部署和微服务调用链的特点，建立微服务系统状态图来刻画微服务的请求流量、系统配置与安全性、服务质量、资源开销之间的关系；其次，设计 D2RA 框架并提出基于深度 Q 网络的动态策略优化算法，为微服务提供动态请求流量下最优系统配置快速更新方案。仿真实验结果表明，D2RA 在动态请求流量下可有效进行资源分配，相对于对比方法在防御有效性和服务质量方面分别取得 19.07%和 42.31%的优化。

关键词：微服务；云原生；动态防御；强化学习；深度 Q 网络
中图分类号：TP309.1
文献标志码：A
DOI: 10.11959/j.issn.1000−436x.2023077

Research on multidimensional dynamic defense strategy for microservice based on deep reinforcement learning

ZHOU Dacheng, CHEN Hongchang, HE Weizhen, CHENG Guozhen, HU Hongchao
Institute of Information Technology, Information Engineering University, Zhengzhou 450002, China

Abstract: Aiming at the problem that it is hard for security defense strategies in cloud native to guarantee the quality of service under dynamic requests, a multidimensional dynamic defense strategy for microservice based on deep reinforcement learning, named D2RA strategy, was proposed to provide dynamic configuration schemes that ensure security defense performance and quality of service for microservices under dynamical requests. Firstly, based on the characteristics of multiple replicas and invocation chains of microservices, a microservices state graph was established to depict the maps between requests, system configuration and security performance, quality of service, and resource overhead of microservices. Secondly, the D2RA framework was designed and a dynamic strategy optimization algorithm based on deep Q-network was proposed for microservices to provide fast and optimal system configurations update scheme under dynamic requests. The simulation results show that D2RA effectively allocate resources under dynamic requests, and achieve 19.07% more defense effectiveness and 42.31% higher quality of service as compared to the existing methods.

Keywords: microservice, cloud native, dynamic defense, reinforcement learning, deep Q-network

0 引言

随着云计算技术的发展，面向云上应用程序的云原生（CN, cloud native）技术生态逐渐形成[1]。在云原生环境中，基于微服务架构[2]的复杂应用程序被解耦为多个功能独立、业务耦合的微服务，其中，单个微服务通常实现单一的业务功能，多个微服务通过轻量级的通信协议形成微服务调用链，共同完成复杂的业务功能。云原生应用基于容器化部署的轻量、自治、松耦合的微服务，支持独立发布

收稿日期：2022−12−04；**修回日期：**2023−02−23
基金项目：国家自然科学基金资助项目（No.62072467）；国家重点研发计划基金资助项目（No.2021YFB1006200, No.2021YFB1006201）
Foundation Items: The National Natural Science Foundation of China (No.62072467), The National Key Research and Development Program of China (No.2021YFB1006200, No.2021YFB1006201)

与迭代的敏捷开发，逐渐成为流行的云上应用程序的交付模式。

微服务架构也使云上应用程序的安全治理变得更加困难。随着云服务业务复杂程度的增大，微服务的规模呈爆炸式增长，例如，阿里巴巴拥有超过 20 000 个微服务[3]。大规模的微服务形成了复杂的微服务调用链，微服务之间的网络通信暴露出更大的攻击面。此外，容器化微服务支持细粒度部署、伸缩与迁移，微服务的部署状态可以随服务需求的变化而动态改变，因此，其攻击面具有高度的时变性。面对规模庞大、动态部署的微服务，入侵检测[4]、分布式授权[5]等传统安全防御技术的实施难度陡增且防御效果甚微。移动目标防御（MTD, moving target defense）[6]动态改变系统的攻击面，为系统增加不确定性因素，干扰攻击行为，增强系统的动态防御能力。MTD 这种动态防御技术给微服务的安全治理提供了新的思路。Torkura 等[7]针对微服务的共有漏洞引发的横向移动攻击，通过动态转换微服务的编程语言与容器镜像来改变微服务的攻击面，提高微服务的安全性。Jin 等[8]针对容器云环境下的微服务建立多维攻击图模型，提出关键微服务的评估方法，并针对关键微服务进行 MTD 策略的优化。张帅等[9]对微服务进行整体防护，基于深度强化学习算法优化微服务的清洗周期来提高系统的安全性。

但是，现有的微服务动态防御策略聚焦在增强系统的安全防御能力，缺乏与资源分配策略的协同，在动态请求流量下难以保证系统的服务质量（QoS, quality of service）。微服务资源分配策略[10]用来在动态请求流量下维护微服务的服务质量，提高资源利用率。具体来讲，微服务中的资源分配算法观测或预测微服务的服务质量的变化[11]，动态优化微服务的系统配置。该过程使微服务系统配置动态改变，易引起防御有效性“漂移”，现有研究[8-9]通过优化动态防御策略的求解速度来提高对动态环境的适应能力。但由于动态防御策略在优化时缺乏对服务质量的考虑，动态防御策略对系统配置的更改可能会引起资源分配策略对系统配置的失效。综上所述，现有的动态防御策略与资源分配策略相互独立地更改微服务系统配置，存在配置策略相互矛盾而反复振荡的问题，对防御效果和服务质量均造成不利的影响。

针对以上问题，本文提出了基于深度强化学习的微服务多维动态防御策略，简称 D2RA 策略，给出在动态请求流量下兼顾安全防御和服务质量的微服务系统配置多维目标优化的解决方案。首先，分析微服务多副本随机负载均衡的部署模式对多阶段攻击的扰乱效果，提出面向微服务的安全性和服务质量的微服务变换和清洗相结合的动态防御策略，抵御攻击者沿着微服务链逐步渗透的横向移动攻击。然后，为达到最佳的配置效果，设计了 D2RA 策略框架，提出了基于深度 Q 网络的动态策略优化（DQN-DSO, deep Q network based dynamic strategy optimization）算法，求解微服务在动态请求流量下的防御效果和服务质量最优的系统配置方案。最后，对 DQN-DSO 算法进行充分的训练，为微服务提供及时有效的动态防御策略。本文的主要研究工作及贡献如下。

1) 建立微服务系统状态图（SSG, system state graph）模型来刻画微服务的请求流量、系统配置与微服务安全性、服务质量和资源开销的关系，提出多副本动态部署的微服务的安全模型来刻画攻击者通过网络逐个入侵微服务调用链过程中的系统风险值；提出基于排队论的微服务调用链的响应时间模型来刻画微服务系统配置对服务质量的影响。同时，提出基于微服务应用程序接口（API, application programming interface）网关的请求流量来求解各个微服务请求到达率的算法。

2) 设计微服务在动态请求流量下的多维动态防御策略更新的 D2RA 策略框架，提出 DQN-DSO 算法，为微服务提供动态请求流量下的副本数量和清洗周期的配置更新方案，提高动态防御的有效性和微服务的服务质量。同时，通过分解 DQN-DSO 算法的动作时间步，解决大规模动态微服务的高维解空间中优化求解困难的问题。

3) 仿真实验结果表明，所提 DQN-DSO 算法具有良好的收敛效果。与现有的微服务的 MTD 策略相比，D2RA 策略能够在动态请求流量的场景下给出有效的系统配置更新决策，在防御有效性和服务质量方面分别提高 19.07%和 42.31%，验证了所提策略的有效性。

1　问题分析

本节基于示例微服务环境，阐述微服务的安全威胁，提出应对微服务环境的动态防御策略，分析实施该策略面临的主要挑战。

1.1 威胁场景

本文采用网络杀伤链（CKC, cyber kill chain）模型[12]来分析攻击者针对微服务节点的攻击过程。在该模型中，一个完整的攻击过程包括前期侦察、漏洞利用、权限获取、安装后门和扩大影响等多个步骤。在云原生环境中，单个微服务运行在独立的容器上，多个微服务通过容器编排形成网络连通的微服务调用链，共同完成复杂的业务功能。攻击者首先探测目标微服务，获取发动攻击的有效信息；然后利用微服务的安全漏洞或容器的安全漏洞投递攻击载荷，获取目标微服务的控制权限及安装后门。在完成一个微服务的渗透之后，攻击者为进一步扩大攻击影响，通过横向移动[13]逐个侵入微服务调用链中的各个微服务，达到侵入目标微服务或获取一定的控守规模的攻击目标[14]。例如，微服务调用链的叶子节点通常是存储数据的微服务节点[3]，可能是攻击者的攻击目标。

本文假设云平台和云服务提供商可信，攻击者从云环境的外部网络发起攻击[8]，攻击者可以利用微服务及其容器的漏洞侵入并劫持微服务，并且基于已劫持的微服务及其容器，可以进一步对网络可达的其他微服务发起攻击[15]。假设运行微服务的容器的网络配置服从最小化原则，只有微服务之间存在调用关系时，运行微服务的容器才网络可达[8]。

微服务开源实验基准的社交网络微服务系统如图 1 所示[16]，其中，数据库 MongoDB 存储微服务关键数据，通常是攻击者的攻击目标。攻击者通过多条攻击路径窃取存储在 MongoDB 的数据，例如，攻击者可以依次渗透 Nginx、Make_Post、URL_Storage、Compose_Post、Post_Storage 和 MongoDB，以达到窃取关键数据的目的。

1.2 多维动态防御策略

MTD 基于多样化、动态化和冗余技术实现不同攻击场景下的动态防御效果[17]。在微服务场景中，高效敏捷的容器云环境放大了系统的动态性和复杂性，动态防御策略的设计应充分考虑云环境下资源的动态分配所带来的影响，从而保证动态防御策略在生产环境下可以更加有效地部署，在防御微服务的威胁的同时，保证微服务的服务质量。本文采用微服务副本变换和副本清洗结合的方式，基于资源动态配置实现多维动态防御策略，扰乱攻击者沿着微服务链逐步渗透的过程。

1) 微服务副本变换。微服务采用多副本部署，并且通过随机负载均衡的方式动态变换对用户提供服务的微服务副本，扰乱攻击者对其先前注入攻击载荷的微服务副本的持续操控，增加攻击者的渗透难度。

2) 微服务副本清洗。无状态的微服务在运行过程中进行动态清洗来改变攻击面[18]，避免攻击者注入的攻击载荷的驻留，切断攻击者沿着微服务调用链进一步渗透的路径。在实施过程中，容器编排平台根据微服务副本清洗周期，基于只读容器镜像逐个创建新的微服务副本，同时将先前的微服务副本逐个下线，即可达到微服务副本清洗的目的。

本文以图 1 所示的微服务场景为例说明上述微服务 MTD 策略的有效性。若微服务 Nginx 的副本 Nginx-1 在上一次的攻击过程中被注入了攻击载荷，攻击者需要继续操控微服务副本 Nginx-1 才能对微服务调用链上被 Nginx 调用的微服务节点 Make_Post 实施攻击。但是，通过微服务的多副本的随机负载均衡，微服务 Nginx 对攻击者呈现随机变换的效果，攻击者需要对微服务 Nginx 进行多次渗透，才能再次锁定微服务副本 Nginx-1 来实施下一步的攻击。在完成多次渗透之前，若微服务 Nginx 的副本 Nginx-1 被清洗，那么攻击者此次攻击行为失败。

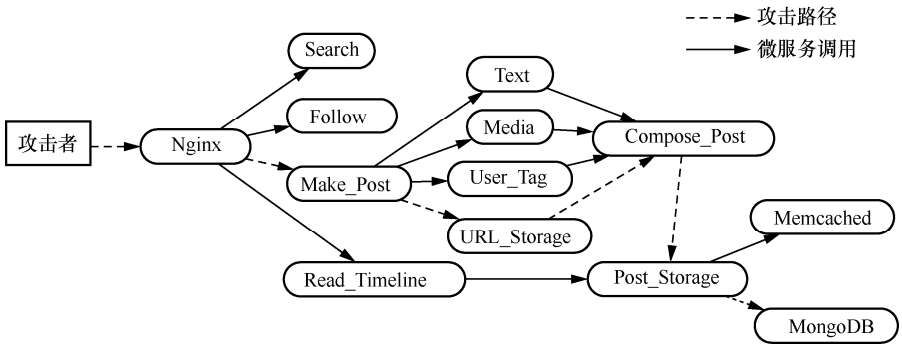


图 1 微服务开源实验基准的社交网络微服务系统

本文采用微服务副本变换和微服务副本清洗这 2 种突变元素来实现微服务系统的动态防御效果。在此过程中，通过优化微服务副本的数量和清洗周期配置达到优化防御性能、保证服务质量的多维目标，该目标通过改变微服务托管平台的资源管理模块即可实现，难度较小。

1.3 主要挑战

多维动态防御策略在微服务场景下主要面临以下挑战。

1) 大规模的微服务使微服务动态防御策略优化的求解复杂度变大。随着微服务规模的增大，针对微服务的优化问题的解空间维度增加，问题求解较困难。如何在大规模的微服务环境下协调资源消耗、服务质量与安全防御的效果，进行动态防御策略优化是需要解决的问题。

2) 云环境的动态性要求系统配置更新具有实时性，这对最优动态防御配置的求解速度提出挑战。云环境的请求流量具有高度的动态性，需动态调整微服务副本的数量来满足系统的服务质量，已部署的动态防御策略易出现有效性“漂移”。如何快速调整微服务的系统配置来保证微服务系统的服务质量和防御效果是需要解决的问题。

2 问题建模

本节通过刻画微服务的请求流量、系统配置与微服务系统的安全性、资源开销和服务质量的关系，形式化描述多维动态防御策略的待优化的问题。

2.1 微服务系统状态图

本文建立微服务系统状态图模型来刻画微服务的请求流量、系统配置与系统指标的关系，在 t 时刻，系统状态图 $\mathrm{SSG}_t=\left(V,E,N_t,T_t,\lambda_t,\varphi_t,\psi_t,\chi_t\right)$，参数说明如下。

1) $V=\{V_1,V_2,\cdots,V_l\}$ 表示 l 个微服务集合，微服务 V_i 包含 n_i 个微服务副本，即 $V_i=\{v^k\mid 1\leqslant k\leqslant n_i\}$，各个微服务副本由同一微服务镜像创建而成，具有相同的功能和资源配置。

2) $E=\{e_{ij}\mid V_i,V_j\in V,e_{ij}\in\{0,1\}\}$ 表示 l 个微服务之间的连边集合，若 $e_{ij}=0$，则表示微服务 V_i 不存在对微服务 V_j 的调用关系，两者网络不可达；若 $e_{ij}=1$，则表示微服务 V_i 存在对微服务 V_j 的调用关系，记微服务 V_i 是微服务 V_j 的上游微服务（UM, upstream microservice），相应地，微服务 V_j 是微服务 V_i 的下游微服务（DM, downstream microservice），在一次请求中，微服务 V_i 存在副本 $v^k\in V_i$ 去调用微服务 V_j 的副本 $v^{k+1}\in V_j$。

3) $\boldsymbol{N}_t=\{n_1,n_2,\cdots,n_l\}$ 表示 l 个微服务分别配置的副本数量配置向量，n_i 表示微服务 V_i 的副本数量。

4) $\boldsymbol{T}_t=\{\tau_1,\tau_2,\cdots,\tau_l\}$ 表示 l 个微服务各自的副本清洗周期配置向量，τ_i 表示微服务 V_i 的每个副本的清洗周期，即 V_i 每隔时间 τ_i 创建 n_i 个副本，并且在新副本创建成功之后删除旧副本。

5) $\lambda_t=\{\lambda_1,\lambda_2,\cdots,\lambda_l\}$ 表示 l 个微服务各自的请求到达率。

6) φ_t 表示微服务系统的安全性，用来评估多维动态防御策略的防御效果。

7) ψ_t 表示微服务系统的响应时间，用来评估微服务系统的服务质量。

8) χ_t 表示微服务系统的资源开销。

2.2 安全量化模型

攻击者侵入运行微服务的容器，进行提权操作获取容器乃至宿主机权限，进而实施横向移动。在此过程中，攻击者需要通过 API 调用，如恶意的 HTTP 请求，对同一个微服务副本发动多次探测与注入才能进行提权操作。对于具有 n_i 个副本的微服务 V_i，输入的 API 请求由随机负载均衡器等概率重定向至 n_i 个微服务副本上，任一微服务 V_i 的副本接收攻击者发出的 API 调用的概率为

$$p_i=\frac{1}{n_i},V_i\in V \tag{1}$$

根据攻击链理论，一次成功的网络攻击通常历经多个步骤[12]，本文假设攻击者需要多次锁定攻击目标（即本文所述场景的微服务的某一个副本）进行攻击，才能达到预期的攻击效果。本文假设攻击者需要 Δ 次尝试对同一微服务副本 $v^k\in V_i$ 实施渗透才能侵入微服务 V_i，记攻击者侵入微服务 V_i 的一个副本需要尝试攻击的次数为 X_i，即前 X_i-1 次需要成功侵入 $\Delta-1$ 次，因此，X_i 的概率分布函数为

$$\Pr\{X_i=x\}=\frac{p_i^{\mathrm{succ}}}{n_i}\mathrm{C}_{x-1}^{\Delta-2}\left(\frac{p_i^{\mathrm{succ}}}{n_i}\right)^{\Delta-2}\left(1-\frac{p_i^{\mathrm{succ}}}{n_i}\right)^{x-\Delta+1}=\left(\frac{p_i^{\mathrm{succ}}}{n_i}\right)^{\Delta-1}\mathrm{C}_{x-1}^{\Delta-2}\left(1-\frac{p_i^{\mathrm{succ}}}{n_i}\right)^{x-\Delta+1},V_i\in V,x\geqslant\Delta \tag{2}$$

其中，p_i^{succ} 为微服务 V_i 单次攻击奏效的概率，在应用中可由基于通用漏洞评分系统（CVSS, common vulnerability scoring system）的评估方法[8]计算得出。本文的研究暂不考虑攻击者的攻击经验积累对攻击实施时间的影响，假设攻击者能力足够强大，每次实施攻击所需的时间可忽略不计。同时，受限于防火墙等传统网络设备的检测能力，攻击者难以隐匿地进行高频次攻击，本文假设攻击者的攻击尝试的时间间隔为 T^{att}。因此，攻击者花费时间 t 来攻击微服务 V_i 且攻击成功的概率为

$$p_i(t)=\Pr\left\{\Delta\leqslant X_i\leqslant\left\lfloor\frac{t}{T^{\text{att}}}\right\rfloor\right\}=\left(\frac{p_i^{\text{succ}}}{n_i}\right)^{\Delta-1}\sum_{x=\Delta}^{x=\left\lfloor\frac{t}{T^{\text{att}}}\right\rfloor}C_{x-1}^{\Delta-2}\left(1-\frac{p_i^{\text{succ}}}{n_i}\right)^{x-\Delta+1},V_i\in V \quad (3)$$

因此，当微服务 V_i 的动态清洗周期为 τ_i 时，微服务被攻击成功的概率为 $p_i(\tau_i)$。由式(3)可知，微服务的动态清洗周期越长，微服务被侵入成功的概率越高。为了降低攻击者攻击成功的概率，需要合理设置微服务的清洗周期。

记多个微服务依次调用的序列为微服务调用链，表示为 $P=\{V_{i_1},V_{i_2},\cdots,V_{i_r}\}$，其中 $d\in[1,r-1]$，$e_{i_d,i_{d+1}}=1$。$V_{i_d}\in V$ 表示一个微服务节点，d 表示微服务 V_i 在微服务调用链 P 中的调用深度。在攻击者针对该微服务调用链进行的攻击中，攻击者需要先从微服务 V_{i_1} 的一个副本开始攻击，在微服务 V_{i_1} 副本被渗透之后，基于微服务 V_{i_1} 的副本再朝着微服务 V_{i_2} 横向移动，直到攻击者侵入微服务调用链的终点 V_{i_t}。对于攻击者而言，即使无法侵入微服务调用链的终点，也可以在针对微服务节点的攻击中对系统造成干扰，因此从防御者的角度无法预测攻击者的攻击目的是哪一个微服务节点。基于以上讨论，本文不以攻击者渗透至最终的攻击目标作为攻击收益的衡量方式[8-9]，而是将所有微服务在微服务的阶段性攻击中被攻击的风险都纳入对微服务系统的安全性量化中。对于攻击者而言，一方面，攻占的微服务节点的数量越多，其攻击收益越大；另一方面，微服务调用链的终点通常为存储数据的微服务[3]，越接近微服务调用链的终点，攻击者的攻击收益越高。因此，攻击者的攻击收益为

$$\varphi_P=\sum_{V_i\in P}\left(1-\mathrm{e}^{-\xi d}\right)p_i\left(\tau_i\right),P\subseteq\text{SSG} \quad (4)$$

其中，$\xi(0<\xi<1)$ 为攻击收益的增长幅度调节因子，ξ 越小，随着微服务调用深度 d 的增加速度越快。式(4)表示随着 d 的增加，攻击者逐渐接近微服务调用链的终点，攻击者的攻击收益逐渐增大。

由于攻击者由外部网络发起攻击，通过任意一条可达攻击目标的微服务调用链都可以展开攻击，攻击成功概率最高的微服务调用链可以用来表示系统的防御能力。在 SSG_t 的配置条件下，微服务系统被攻击成功的概率可表示为

$$\varphi_t=\max_{P\subseteq\text{SSG}}\{\varphi_P\} \quad (5)$$

式(5)可衡量多维动态防御策略针对微服务系统的防御能力，其值越小，表明微服务系统的风险值越低，多维动态防御策略的防御能力越强。本文使用 φ_t 来表征微服务系统的风险值，在 t 时刻通过最小化 φ_t 值来搜索最佳的多维动态防御策略。

2.3 响应时间模型

本文所设计的多维动态防御策略在降低系统被攻击成功的风险的同时，需要保证系统的服务质量。微服务的服务响应时间是面向用户的应用程序的服务质量，一般来讲，过长的响应时间会引起 QoS 的下降，影响用户体验及商业利益。据统计，亚马逊网站的响应时间每增加 100 ms，就产生 1% 的销量下跌[19]。排队模型常用来刻画微服务的响应时间模型[20-21]，本文将多副本微服务建模为 M/M/N 排队系统。记微服务 V_i 的单个副本的服务率为 μ_i，则响应时间可以表示为

$$\text{RT}_i=\frac{1}{\lambda_i}\left(\frac{\lambda_i}{\mu_i}+\rho_i\frac{\left(n_i\rho_i\right)^{n_i}}{n_i!}\frac{\pi_0^i}{\left(1-\rho_i\right)^2}\right)^{-1},V_i\in V \quad (6)$$

其中，$\rho_i=\dfrac{\lambda_i}{n_i\mu_i},0\leqslant\rho_i<1$，$\pi_0^i$ 表示微服务 V_i 的队列中无请求到达的稳态概率，即

$$\pi_0^i=\left[\sum_{r=0}^{n_i-1}\frac{\left(n_i\rho_i\right)^r}{r!}+\frac{\left(n_i\rho_i\right)^{n_i}}{n_i!\left(1-\rho_i\right)}\right]^{-1},V_i\in V \quad (7)$$

对于来自微服务的 API 网关（即图 1 中的 Nginx）的用户请求，微服务将会以一定的概率分发给其 DM[22]，例如，Nginx 发出的请求将会分发至 Search、Follow、Make_Post 和 Read_Timeline 这 4 个微服务。记微服务 V_i 的 UM 集合为 $\text{UV}_i=\{V_u|e_{ui}=1,V_u\in V\}$，DM 集合为 $\text{DV}_i=\{V_d\mid e_{id}=1,V_d\in V\}$，微服务 V_i 接收来自 UM 的所有调用请求，并且对所

有 DM 发起调用请求。假设微服务 V_i 在调用 DM 时，仅转发来自 UM 的请求，则微服务调用请求的数量满足

$$\sum_{V_u \in \mathrm{UV}_i} \lambda_{ui} = \sum_{V_d \in \mathrm{DV}_i} \lambda_{id} \tag{8}$$

其中，λ_{ij} 表示微服务调用连边 e_{ij} 通过的请求到达率，记为连边请求到达率（RTE, request through edge）。本文假设微服务 V_i 对多个 DM 等概率调用，则有

$$\lambda_d = \sum_{e_{id} \in E \wedge e_{id}=1} \frac{\lambda_i}{\left|\mathrm{DV}_i\right|}, 1 \leqslant i,d \leqslant |V| \tag{9}$$

记用户的请求到达率为 λ，则微服务系统状态图 SSG_t 的根节点（如图 1 中的 Nginx）的请求到达率 $\lambda_1 = \lambda$，基于此，可迭代求取微服务图 G 的任一微服务节点 $V_i \in V$ 的请求到达率 λ_i。式(9)求取 DM 的请求到达率需要遍历其所有的 UM 的请求到达率 λ_i 及 UM 的 DM 个数 $\left|\mathrm{DV}_i\right|$，具体求解过程如算法 1 所示。首先，初始化待搜索微服务节点队列 Q 和各个微服务已计算 RTE 的上游连边计数表 Count（第 1)行)。上游连边计数表 Count 的作用是避免遗漏调用微服务 V_d 的 UM 给微服务 V_d 引入的 RTE。同时，将 SSG_t 的根节点 SSG.root 添加至待搜索节点队列 Q 中，以保证搜索从根节点开始（第 2)行)。在待搜索微服务节点队列 Q 非空时（第 3)行)，循环从 Q 中出栈得出当前搜索节点 u（第 4)行)，若当前搜索节点的 DM 为空，则跳过该节点（第 5)行)，否则计算各个 UM 调用的所有 DM 的连边的 RTE（第 6)～7)行)，每一个 DM 都被添加至待搜索节点队列 Q（第 8)行)，参与后续递归的搜索；当微服务 V_d 的调用计数 Count_d 等于其 UM 的个数时，表明微服务节点 V_d 的所有上游连边的 RTE 都已被遍历计算（第 9)行)，微服务节点 V_d 的请求到达率 λ_d 才得以计算（第 10)行)。算法循环跳出时，待搜索微服务节点队列 Q 为空，即得到各微服务请求到达率 $\lambda_t = \{\lambda_1, \lambda_2, \cdots, \lambda_l\}$。

算法 1　各个微服务请求到达率求解算法

输入　微服务 API 网关请求到达率 λ

输出　各微服务请求到达率 $\lambda_t = \{\lambda_1, \lambda_2, \cdots, \lambda_l\}$

1) Q = Queue.Init ; Count = $\{0\} \times \left|G.\mathrm{node}\right|$
2) $\lambda_{G.\mathrm{root}} = \lambda$; Q.append(SSG.root) ; $\mathrm{Count}_{G.\mathrm{root}} = 1$
3) while Q != $\varnothing$ do
4) u = $Q(0)$; Q.pop(0)
5) if $\left|\mathrm{DV}_u\right|$==0 continue
6) for each V_d in DV_u do
7) $\lambda_{ud} = \frac{\lambda_u}{\left|\mathrm{DV}_u\right|}$; Count_d ++
8) Q.append(V_d)
9) if Count_d == $\left|\mathrm{UV}_d\right|$ do
10) $\lambda_d = \sum_{e_{id} \in E \wedge e_{id}=1} \lambda_{id}$
11) end if
12) end for
13) end if
14) end while

基于算法 1 求得的各个微服务的请求到达率，微服务调用链 $P \subseteq \mathrm{SSG}$ 上各个微服务节点的响应时间可根据式(6)求取。微服务调用链 P 的总响应时间可以表示为微服务调用链上各个微服务的时间之和。微服务系统中响应时间最大的微服务调用链表征了微服务的系统时间，可表示为

$$\psi_t = \max_{P \subseteq \mathrm{SSG}} \left\{ \sum_{V_i \in P} \mathrm{RT}_i \right\} \tag{10}$$

ψ_t 越小，表明微服务系统的服务质量越高。在云计算环境下，通常用户与服务提供商协商的服务等级目标（SLO, service level objective）用来约束系统的服务质量，即系统的响应时间不应超过响应时间服务等级目标 ψ_{SLO}，否则将会给系统服务质量带来不利影响。

2.4　资源开销模型

微服务的资源开销主要来自微服务的多副本同时部署所占用的资源和微服务的动态清洗过程中占用的资源。为简化处理，本文不区分微服务占用的计算资源和存储资源，且假设同一微服务的各个副本占用的资源相同，均为 R_i，则微服务的资源占用量随着微服务副本数量的增加而线性增加。此外，由于微服务的动态清洗过程需要先启动一个新的微服务副本，待新的微服务副本正常运行后，才能将旧的微服务副本下线。因此，在微服务副本动态清洗的过程中，该微服务副本占用了双倍的资源。微服务 V_i 的任一副本的清洗周期为 τ_i，创建一个微服务 V_i 的容器副本时间消耗为 T_i^{stt}，微服务在 SSG_t 配置下所占用的资源总量可表示为

$$\chi_t = \sum_{v_i \in V} n_i R_i \left(1 + \frac{T_i^{\mathrm{stt}}}{\tau_i}\right) \tag{11}$$

χ_t 值越小，表明微服务系统的资源开销越小，本文在优化多维动态防御策略的同时需要尽可能朝着最小化的方向来优化该指标，以最少的资源换取最佳的多维动态防御策略将会提高云计算服务提供商的资源利用率。

2.5　问题描述

本文所设计的 D2RA 策略需要在 t 时刻求解最优的微服务副本数量配置向量 $\boldsymbol{N}_t=\{n_1,n_2,\cdots,n_l\}$ 和微服务副本清洗周期配置向量 $\boldsymbol{T}_t=\{\tau_1,\tau_2,\cdots,\tau_l\}$ 来得到最优的微服务系统状态图 SSG_t^*，在满足响应时间要求的前提下，以最少的资源占用量达到最优的防御效果，优化问题为

$$\begin{aligned}&\min_{\boldsymbol{N}_t,\boldsymbol{T}_t}\{\varphi_t,\psi_t,\chi_t\}\\&\text{s.t. }\psi_t\leqslant\psi_{\mathrm{SLO}},\forall V_i\in V,\tau_i\geqslant T^{\mathrm{att}},n_i\geqslant 1\end{aligned}\tag{12}$$

该问题的搜索空间庞大且要求能够快速求解。一方面，当微服务副本数量为 l 时，为搜索最佳的多维动态防御策略，需要在 $2l$ 维空间中搜索最优解，求解难度较高。另一方面，SSG_t^* 随着用户的请求到达率的输入而动态调整，需要加快求解的速度。

3　D2RA 策略设计

本节设计在动态请求流量下进行微服务系统配置动态更新的 D2RA 策略框架，提出基于深度 Q 网络（DQN, deep Q-network）的动态策略优化算法。

3.1　D2RA 策略框架

本文基于强化学习中智能体与环境交互的模式设计了 D2RA 策略，其框架如图 2 所示，主要包括状态监控模块、DQN-DSO 算法、策略部署模块。

1) 状态监控模块通过微服务的 API 网关监控用户请求流量，并基于算法 1 计算各个微服务的请求到达率 $\lambda_t=\{\lambda_1,\lambda_2,\cdots,\lambda_l\}$，同时根据微服务当前的副本数量配置向量 $\boldsymbol{N}_t=\{n_1,n_2,\cdots,n_l\}$ 和清洗周期配置向量 $\boldsymbol{T}_t=\{\tau_1,\tau_2,\cdots,\tau_l\}$，为基于 DQN 的动态策略优化算法提供输入参数。

2) DQN-DSO 算法从环境中监控得到微服务状态，基于强化学习模型求解当前请求流量下微服务的最优系统配置，即满足式(12)的微服务副本向量 $\boldsymbol{N}_t=\{n_1,n_2,\cdots,n_l\}$ 和微服务动态清洗周期配置向量 $\boldsymbol{T}_t=\{\tau_1,\tau_2,\cdots,\tau_l\}$。

3) 策略部署模块实现微服务横向扩/缩容和微服务副本的清洗功能，将计算得到的最佳的微服务副本配置向量和微服务动态清洗周期配置向量通过云资源管理平台部署至微服务环境中。

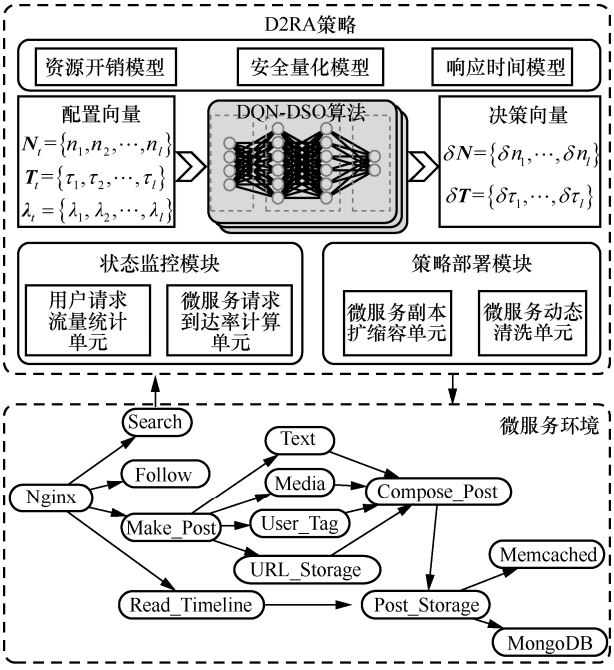


图 2　D2RA 策略的框架

为保证微服务副本清洗周期策略部署的时效性，避免微服务的动态清洗周期配置向量在连续的请求流量下的决策发生冲突，本文记录微服务副本清洗操作的时序数据，并且在做出决策的同时给出是否进行清洗策略覆盖的决策，即是否以当前的清洗策略覆盖先前的清洗策略。具体来讲，若新的清洗周期大于旧的清洗周期，则当前未完成的清洗周期以新的清洗周期为准；否则立即执行清洗操作。综上，约束微服务 V_i 的下一次执行清洗策略的时刻 t_i^s 为

$$t_i^s=\begin{cases}t_i^{s-1}+\tau_i, & t<t_i^{s-1}+\tau_i\\ t, & t\geqslant t_i^{s-1}+\tau_i\end{cases}\tag{13}$$

其中，t 为当前时刻，即策略下发时刻；t_i^{s-1} 为微服务最近一次执行清洗策略的时刻。

3.2　基于 DQN 的动态策略优化算法

本文基于深度学习与强化学习结合的深度 Q 网络[23]对问题进行求解，提出 DQN-DSO 算法。DQN 通过 Q 网络估计状态下的动作价值，避免出现传统 Q 学习算法在高维状态–动作空间引发的 Q 表爆炸的问题。首先，构建马尔可夫决策过程（MDP, Markov decision process）如下。

1) 状态 s_t。基于微服务系统状态图 SSG_t 提取状态信息，包含当前的微服务的副本个数配置向量 $\boldsymbol{N}_t=\{n_1,n_2,\cdots,n_l\}$、微服务的清洗周期配置向量 $\boldsymbol{T}_t=\{\tau_1,\tau_2,\cdots,\tau_l\}$、微服务用户请求到达率配置向量 $\boldsymbol{\lambda}_t=\{\lambda_1,\lambda_2,\cdots,\lambda_l\}$。状态 s_t 可表示为

$$s_t=\{\boldsymbol{N}_t,\boldsymbol{T}_t,\lambda_t\} \tag{14}$$

2) 动作 a_t。策略部署模块需要在当前工作负载 $\lambda_t=\{\lambda_1,\lambda_2,\cdots,\lambda_n\}$ 下决定各个微服务副本数量改变决策向量 $\delta\boldsymbol{N}=\{\delta n_1,\delta n_2,\cdots,\delta n_l\}$ 以及清洗周期改变决策向量 $\delta\boldsymbol{T}=\{\delta\tau_1,\delta\tau_2,\cdots,\delta\tau_n\}$，这对策略部署模块来说是高维度决策。高维的动作空间使不同状态动作间的 Q 值估计的运算量过大，造成 DQN 难以收敛的问题。为解决该问题，本文将高维度单步决策分解为多次低维度决策。低维度决策包括选定一个待更新微服务节点、改变选定微服务节点的副本数量、改变选定微服务节点的清洗周期。动作空间表示为

$$A=\{i^+,i^-,\delta N^+,\delta N^-,\delta T^+,\delta T^-,0\} \tag{15}$$

各个动作的含义如表 1 所示。在训练的过程中，经过多次迭代对动作空间中的动作进行重复与组合，得到当前时刻满足式(12)约束的决策向量 $\{\delta\boldsymbol{N},\delta\boldsymbol{T}\}$。在模型训练过程中，经过最大化累计收益逐步优化每一个请求流量下的决策。

表 1　　各个动作的含义

动作	含义
i^+	选定微服务的索引增加 1
i^-	选定微服务的索引减少 1
δN^+	选定微服务的副本增加一个
δN^-	选定微服务的副本减少一个
δT^+	选定微服务的副本清洗周期增加 1 s
δT^-	选定微服务的副本清洗周期减少 1 s
0	不采取任何动作

3) 收益 r_t。式(12)定义的多目标优化问题是强化学习算法求解的任务目标，由 φ_t、ψ_t、χ_t 这 3 个指标构成。本文基于动态权重法[24]将多目标动态优化问题转化为单目标动态优化问题，以得到工程落地中所需要的确定性决策。此外，为了在强化学习中求解式(12)最小化的优化目标，本文首先对 φ_t、ψ_t、χ_t 进行取倒数的处理；其次，对各个指标进行归一化处理，以保证单个指标对目标的影响相对公平；最后，根据指标对求解问题的重要程度赋予响应的权重。因此，待优化的目标函数为

$$O_t=\omega_1\vartheta_1\left(\frac{1}{\varphi_t}\right)+\omega_2\vartheta_2\left(\frac{1}{\psi_t}\right)+\omega_3\vartheta_3\left(\frac{1}{\chi_t}\right) \tag{16}$$

其中，ω_1、ω_2、ω_3 为指标的权重系数，ϑ_1、ϑ_2、ϑ_3 为指标的归一化函数。此外，通过惩罚因子 penalty 将式(12)的约束条件纳入收益函数中。综上，收益函数表示为

$$r_t=O_t-\mathrm{penalty} \tag{17}$$

基于上述 MDP 模型，本文的 Q 网络首先采用卷积神经网络（CNN，convolutional neural network）提取 $3\times l$ 的状态信息，接着通过全连接网络将 CNN 的输出张量映射到动作空间中。

在训练过程中的样本探索阶段，DQN 根据 $Q(s,a,\theta)$ 网络输出 Q 值的估值，基于“ε-贪婪策略”对选择将执行的动作，即以 $1-\varepsilon$ 的概率选择当前状态下 Q 值的估值最大的动作，以 ε 的概率随机选择动作。随着训练的进行，ε 从 $\varepsilon_{\max}$ 开始，分 M 步线性下降至 $\varepsilon_{\min}$，协调 DQN 对未知动作的“探索”能力与对已学习到的动作的“利用”能力的关系，使训练前期更加广泛地探索未尝试过的动作，训练后期更加充分地利用训练所得的经验。此外，为提高样本利用率，DQN 使用经验回放池 D 存储样本 (s,a,r,s')。当 D 中样本数量达到批处理要求后，DQN 开始训练 Q 网络，使其能够更加精确地估计状态动作价值估计，即最小化误差函数

$$J(\theta)=\mathrm{E}_{(s,a,s',r)\in D}\left[\frac{1}{2}\big(y(s,a)-Q(s,a,\theta)\big)^2\right] \tag{18}$$

其中，$y(s,a)$ 由观测到的当前时刻的收益 $r(s,a)$ 和 $Q(s',a',\theta^-)$ 网络估计的下一时刻的估值构成，即

$$y(s,a)=r(s,a)+\gamma\max_{a'}Q\left(s',a',\theta^-\right) \tag{19}$$

其中，γ 为下一时刻收益的折扣因子，$Q\left(s,a,\theta^-\right)$ 为与 $Q(s,a,\theta)$ 结构相同的目标 Q 网络。算法每迭代 C 次进行一次网络参数的更新 $\theta^-\leftarrow\theta$，其目的为固定一段时间内的 Q 网络的参数，避免训练过程中 Q 网络的波动，提高模型的稳定性。

本文使用图 3 所示的美国电子零售商的在线流量数据作为每个回合（episode）的动态请求到达率的数据集，记为 Θ。具体训练过程如算法 2 所示。

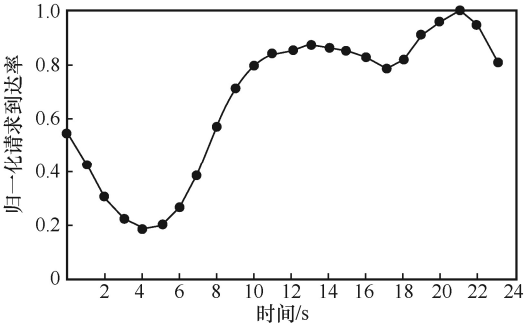


图 3 美国电子零售商的在线流量数据

算法 2 DQN-DSO 算法训练过程

输入 微服务图 G，流量数据集 Θ

输出 Q 网络参数 θ

初始化经验回放缓冲区 D，Q 网络参数 θ，目标 Q 网络参数 θ^-，网络更新步长 L，$\varepsilon \leftarrow \varepsilon_{\max}$
for episode in STEP
  随机生成微服务初始配置 $\{N_0, T_0\}$
  for $\lambda \in \Theta$ do
    基于算法 1 计算各微服务请求率 λ_t
    while $\{\delta \boldsymbol{N}, \delta \boldsymbol{T}\}$ 未满足式(12)约束 do
      生成状态 $s_t = \{\boldsymbol{N}_t, \boldsymbol{T}_t, \lambda_t\}$
      $\varepsilon \leftarrow \varepsilon - \dfrac{\varepsilon_{\max} - \varepsilon_{\min}}{M}$
      if $\varepsilon < \varepsilon_{\min}$ do
        $\varepsilon \leftarrow \varepsilon_{\min}$
      end if
      if random(0,1) < ε do
        随机选择动作 $a_t \in A$
      else do
        $a_t \leftarrow \max_{a'} Q(s', a', \theta^-)$
      end if
      执行动作 a_t 得到下一状态 s_{t+1}
      基于式(17)计算收益 r_t
      将样本 $< s_t, a_t, r_t, s_{t+1} >$ 存入 D
      if $|D| >$ batch do
        从 D 中采集 batch 个样本，基于式(18)计算误差函数 $J_Q(\theta)$，使用梯度下降法更新 θ
      end if
      if $|D| \% L == 0$ do
        $\theta^- \leftarrow \theta$
      end if
    end while
  end for
end for

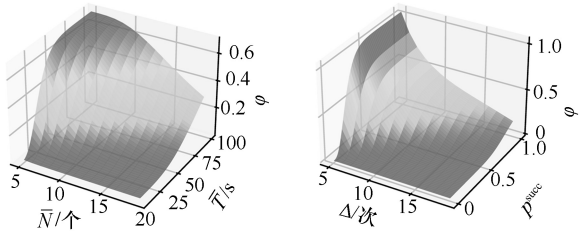
4 仿真与评估

本节通过仿真实验，验证 D2RA 策略中基于 DQN 的动态策略优化算法的收敛性；评估微服务系统在动态工作负载下的安全性、服务质量和资源开销，验证 D2RA 策略的有效性。

4.1 仿真设置

仿真实验平台为一台 CPU 配置为 48 核心的 Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50 GHz、RAM 配置为 32GB DDR、磁盘空间配置为 500 GB 的 NSFocus MICA-9700 服务器。DQN-DSO 算法基于 Pytorch 1.12.1 实现，语言版本为 Python 3.9。神经网络由两层二维 CNN 和两层全连接层构成，其中两层 CNN 的输入通道分别为 1、16，输出通道分别为 16、3，卷积核尺寸为 5；两层全连接层输入神经元个数分别为 $32\times3\times l$、64，输出神经元个数分别为 64、7；各层之间的激活函数均为 ReLU。本节基于 networkx 2.8.4 工具包实现图 1 所示的微服务拓扑，并使用美国电子零售商的在线流量数据模拟微服务的动态配置的场景。在仿真实验中，在请求流量数据中的每一时刻，D2RA 策略基于当前微服务的系统状态进行一次微服务系统配置的优化与更新，每次优化求解中 DQN-DSO 算法迭代多次得出有效决策。

为合理设置模型参数，本文对微服务的安全模型进行关键参数分析，以指导仿真实验的关键参数设置。关键参数对微服务系统风险值的影响如图 4 所示。



(a) 平均副本数量与平均清洗周期　(b) 攻击所需次数与单次攻击成功概率

图 4 关键参数对微服务系统风险值的影响

微服务的平均副本数量 $\bar{N}$ 和平均清洗周期 $\bar{T}$ 对微服务系统风险值的影响如图 4(a)所示。系统的风险值随着微服务副本数量的增加而降低，随

着清洗周期的增大而升高，曲面图与预期相符。随着微服务副本数量的增加，通过随机负载均衡可以给攻击者带来更大的目标不确定性，本文假设每个微服务节点最大的副本数量为 20 个。微服务副本的清洗周期越长，则攻击者注入的攻击载荷驻留的时间越久，微服务被攻击成功的风险值越高，本文假设微服务最小的清洗周期为 10 s。微服务副本的攻击所需次数 Δ 与单次攻击成功概率 p^{succ} 对微服务系统风险值的影响如图 4(b)所示。微服务系统风险值随着 Δ 的增大而降低，随着 p^{succ} 的增大而升高。基于仿真结果，结合文献[8-9]的配置，本文设置各个微服务节点任一副本的 p^{succ} 为[0.3, 0.5]的随机值，设置各个微服务节点任一副本的 Δ 为[5, 15]的随机整数。仿真参数设置如表 2 所示。

表 2　　仿真参数设置

仿真参数	取值
单次攻击成功概率 p_i^{succ}	[0.3,0.5]的随机值
攻击所需次数 Δ	[5,15]的随机整数
攻击时间间隔 T^{att} /s	0.5
攻击风险调节系数 ξ	0.8
各个微服务副本的服务率 μ	0.2
微服务响应时间服务等级目标 ψ_{SLO} /s	0.5
微服务安全性目标权重 ω_1	0.4
微服务服务质量目标权重 ω_2	0.3
微服务资源开销目标权重 ω_3	0.3
收益折扣因子 γ	0.9
采样批量 batch	32
学习率	0.000 1
目标网络更新步长 C	200
初始探索概率 $\varepsilon_{\max}$	1
最终探索概率 $\varepsilon_{\min}$	0.1
ε 下降所需步数 M	30 000
经验回放池大小	50 000

4.2　对比方法

在仿真实验中，将 D2RA 与以下策略进行对比。

1) DSEOM[8]。DSEOM 针对容器云环境下的微服务建立多维攻击图模型刻画微服务的攻击难度，指导 MTD 策略的部署。该策略通过计算关键微服务，并针对关键微服务进行防御策略优化。

2) SmartSCR[9]。SmartSCR 将所有微服务节点都作为安全防御目标，使用深度强化学习算法全局优化微服务的动态清洗周期。该策略仅针对微服务副本清洗周期做策略优化，随着资源分配策略调整微服务副本数量，进一步调整各个微服务的清洗周期。

3) D2PSO。该方法为本文所提 D2RA 的简化版本，即通过粒子群优化（PSO）算法求解当前请求流量下的最佳微服务副本数量和微服务副本清洗周期的系统配置。

由于 DSEOM 和 SmartSCR 均只对微服务的安全配置做更新，本文在仿真实验中模拟 Kubernetes 容器编排平台默认的横向自动扩缩容（HPA）方法对微服务的副本数量进行动态扩展，在仿真实验中根据动态请求负载下的资源开销情况进行独立的微服务副本数量的更新。

4.3　结果与分析

4.3.1　DQN-DSO 算法收敛效果

经验回放池大小 $|D|$ 对累计收益值的影响如图 5 所示。从图 5 可以看出，在不同的经验回放池 D 的设置下，DQN-DSO 算法均能在 1 000 回合训练收敛。在训练前期，累计收益值为负，说明算法在随机探索的过程中存在大量违反式(12)约束条件的动作。$|D|$ 影响 Q 网络的稳定性，过小的经验回放池在探索阶段存储的样本较少，缺少足够的经验样本易引起算法收敛不稳定，DQN-DSO 算法收敛之后的累计收益值仍然存在波动，如 $|D|=10\,000$、$|D|=30\,000$ 时，累计收益值分别在 820 回合、1 300 回合存在明显波动。该波动引发累计收益值为负值，说明动态请求流量下依然存在不满足约束条件的动作。而 $|D|=50\,000$ 与 $|D|=70\,000$ 条件下的累计收益值趋于稳定，且两者没有明显区别，因此本文设置 $|D|=50\,000$。

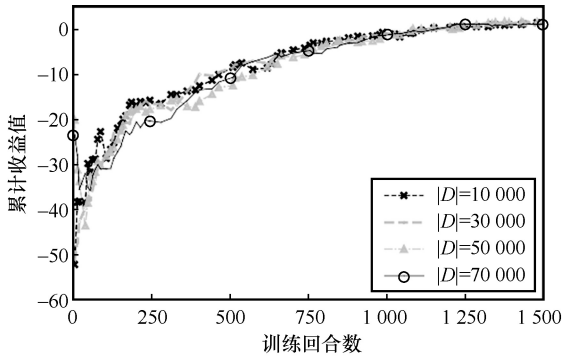


图 5　$|D|$ 对累计收益值的影响

ε 由 $\varepsilon_{\max}$ 下降至 $\varepsilon_{\min}$ 需要的步数 M 对累计收益值的影响如图 6 所示。从图 6 可以看出，随着 M 的减小，DQN-DSO 算法的累计收益值的收敛速度变快，但 $M=5\,000$ 和 $M=10\,000$ 最终的累计收益值明显低于 $M=30\,000$ 和 $M=50\,000$。其原因在于，探索概率 ε 负责协调 DQN 对未知动作的“探索”与对已学习到动作的“利用”的关系。ε 的下降所需步数反映着训练过程中由“探索”到“利用”的转变速度。该转变速度越快，则算法对已学习到动作的利用程度越高，因此可以快速地收敛至最优的动作价值中，但由于训练过程对未知动作的“探索”不足，易使算法收敛于次优解。因此，基于实验结果，本文设置 $M=30\,000$，其收敛速度相对 $M=50\,000$ 的设置较快，且累计收益收敛值与 $M=50\,000$ 的设置无明显差别。

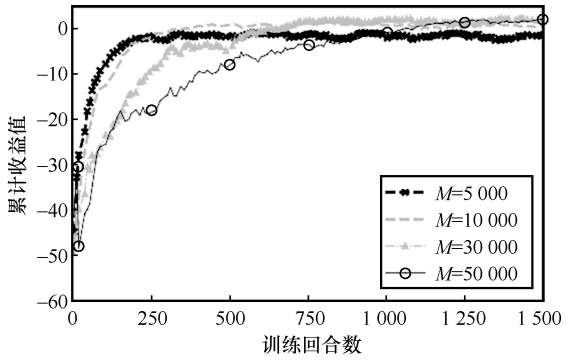


图 6　M 对累计收益值的影响

4.3.2　D2RA 有效性

本文将图 3 所示的请求流量划分为 0 时至 8 时的低峰期、9 时至 17 时的平峰期和 18 时至 23 时的高峰期 3 种不同的请求流量强度等级，通过多次重复实验，分别对微服务的防御能力、响应时间和资源开销情况进行分析。

为衡量 D2RA 对微服务系统的防御效果，本文对比了微服务系统采用不同策略进行系统动态配置时的系统相对风险值，结果如图 7 所示。从图 7 可以看出，在低峰期、平峰期和高峰期，D2RA 的系统风险值相对于其他策略平均降低 9.34%、19.95%和 27.91%，说明 D2RA 具有更强的微服务防御能力。随着流量高峰期的到来，D2RA 的优势更加明显，说明 D2RA 针对微服务的资源配置和防御配置进行综合决策可以更好地应对请求流量的变化。DSEOM 的微服务的系统风险值最大，其主要原因是 DSEOM 针对关键微服务进行防护，存在攻击者绕过关键微服务对关键数据节点进行入侵的可能性，因此，在本文所述的微服务系统风险评估模型下具有较差的表现。SmartSCR 相对于 DSEOM 具有明显的安全性提升，其原因在于 SmartSCR 通过对所有的微服务都进行防护，提高了微服务系统的安全性。D2PSO 和 D2RA 的防御效果更好的原因在于，其综合考虑了资源分配过程中所引入的副本随机化给微服务系统带来的安全增益，因此在动态请求流量下通过针对性的资源分配可以进一步增强微服务系统的安全性。但是，由于粒子群算法搜索最优解过程中存在因早熟而陷入次优解的可能性，D2PSO 在平峰期和高峰期的防御效果均劣于在训练过程中充分探索决策样本的 D2RA。

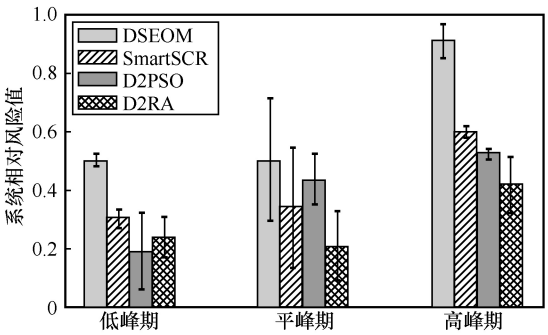


图 7　不同策略下微服务系统相对风险值对比

为衡量 D2RA 在提高系统防御能力的同时对服务质量的影响，本文对比了不同强度的流量负载下微服务分别基于 DSEOM、SmartSCR、D2PSO 和 D2RA 进行系统动态配置的响应时间，结果如图 8 所示，其中响应时间数据均采集于各策略对系统配置进行调整的时刻。从图 8 可以看出，在低峰期、平峰期和高峰期，D2RA 的微服务系统的响应时间相对于其他策略平均降低 9.51%、50.54%和 66.87%。DSEOM 和 SmartSCR 在平峰期和高峰期的响应时间均较高的原因如下：一方面，安全防御策略的部署更改了微服务系统配置，而资源配置策略与安全防御策略相互独立，尚未对系统的资源配置进行更新，导致当前资源配置下的微服务不能对当前的请求流量做出有效的处理；另一方面，默认的动态资源分配策略 HPA 根据资源占用量进行微服务副本数量的调整，在动态请求流量下具有被动性，无法提供及时有效的微服务副本数量配置。此外，由于 SmartSCR 相对于 DSEOM 对所有微服务的配

置都进行更改，对系统资源配置的影响更剧烈，因此其响应时间相对更高，且在高负载情况下的 SLO 违规尤其严重。相对而言，D2RA 及其简化版 D2PSO 在不同请求流量强度下均具有较好的响应时间表现，说明安全策略与资源分配策略综合决策可以在提高系统安全性的同时，保证系统的服务质量。

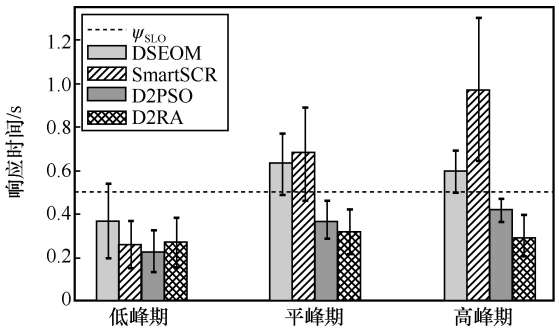


图 8　不同策略下微服务系统响应时间对比

为衡量本文提出的 D2RA 的资源开销情况，本文对比了微服务系统在不同的请求流量强度下采用不同防御策略时的资源开销，结果如图 9 所示。从图 9 可以看出，在低峰期、平峰期和高峰期，D2RA 的微服务系统的资源开销相对于其他策略平均增加 26.24%、39.20%和 30.76%。D2RA 资源增加的原因在于分配了更多的资源来提升系统的安全性和服务质量。在高负载下，D2RA 和 D2PSO 的部署造成微服务系统资源增加的现象尤其明显，其原因在于 D2RA 在请求流量高峰期下为了保证响应时间不超过服务等级目标，分配了更多的微服务副本来保证响应时间。

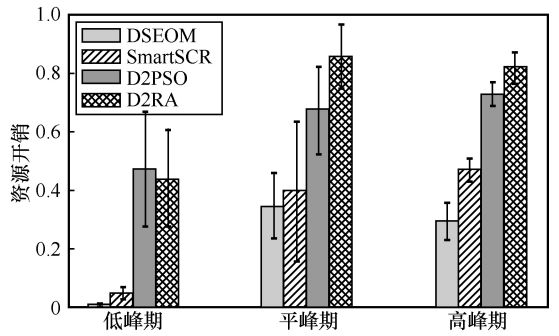


图 9　不同策略下微服务系统资源开销对比

综合图 7～图 9 可知，D2RA 在面对动态请求流量时，通过动态分配更多的资源，在进行有效安全防御的同时满足微服务系统的服务质量，在防御有效性和服务质量方面相对于其他策略平均提高 19.07%和 42.31%。对云租户而言，部署微服务架构下的云原生应用程序时采用 D2RA 策略，可以通过增加租用资源，即增加 32.7%的租用经济成本，获取其所提供的云服务的 19.07%的安全增益和 42.31%的服务质量增益，进而为用户提供更好的服务体验。

4.3.3　算法决策时间对比

为衡量 D2RA 在动态请求流量下的决策速度，本文在仿真过程中分别记录了不同策略在一个动态请求流量点上生成系统配置的决策时间，不同策略在动态负载下的决策时间对比如图 10 所示。

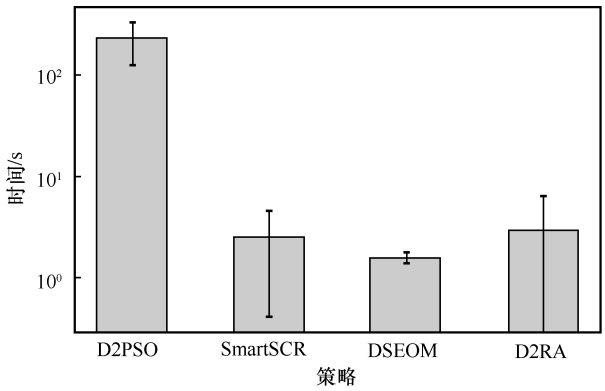


图 10　不同策略在动态负载下的决策时间对比

总体而言，D2RA 的决策时间相对于 SmartSCR 和 DSEOM 略微增长，相对于 D2PSO 算法明显缩短。D2RA 决策时间相对于 SmartSCR 和 DSEOM 较长，其原因如下：一方面，D2RA 的 DQN-DSO 算法在降低动作空间的维度时将动作分解为多个低维动作，在每一个动态流量点的决策都需要 DQN-DSO 算法进行多次迭代；另一方面，对微服务的副本数量和清洗周期同时优化计算，其解空间的维度是 SmartSCR 的 2 倍。DSEOM 由于只对关键微服务节点进行防御策略优化，且优化了最短路径的求解算法，因此其决策时间最短。SmartSCR 基于 Dijkstra 算法求解微服务图中的最短路径，且仅需要对微服务的清洗周期作优化，因此决策时间也相对较短。作为 D2RA 的简化版本，D2PSO 基于 PSO 算法对高维的系统配置决策向量作优化求解，需要设置较多的“粒子”数量进行搜索，消耗的时间呈指数增长，因此，D2PSO 仅适用于请求流量相对静态的场景。

5 结束语

本文针对动态流量下的微服务动态防御策略难以兼顾服务质量的问题展开研究。首先，针对云原生环境下的微服务多副本部署和相互调用的特点，分析微服务多副本动态变换的部署模式对多阶段攻击的扰乱效果，面向微服务的安全性和服务质量，提出基于微服务副本变换和副本清洗的多维动态防御策略。然后，基于微服务系统状态图刻画微服务的请求流量、系统配置与系统指标的映射关系，建立系统的安全量化模型、响应时间模型和资源开销模型，并提出满足防御能力、服务质量和资源开销的多目标动态优化问题。最后，设计动态流量视图下 D2RA 策略框架，并提出基于深度 Q 网络的动态策略优化算法，对动态优化问题进行求解，进而解决大规模动态微服务场景下的动态高维度决策问题求解困难的问题，为动态请求流量下的微服务提供快速的配置更新方案。仿真实验结果表明，相比对比策略，D2RA 在动态流量场景下可有效地进行资源分配并获得更好的防御效果和服务质量，从而能够有效应对动态请求流量下的微服务的安全防御与服务质量难以兼顾的问题。

参考文献：

[1] GANNON D, BARGA R, SUNDARESAN N. Cloud-native applications[J]. IEEE Cloud Computing, 2017, 4(5): 16-21.

[2] THÖNES J. Microservices[J]. IEEE Software, 2015, 32(1): 116.

[3] LUO S T, XU H L, LU C Z, et al. Characterizing microservice dependency and performance: Alibaba trace analysis[C]//Proceedings of the ACM Symposium on Cloud Computing. New York: ACM Press, 2021: 412-426.

[4] NIFE F N, KOTULSKI Z. Application-aware firewall mechanism for software defined networks[J]. Journal of Network and Systems Management, 2020, 28(3): 605-626.

[5] BÁNÁTI A, KAIL E, KARÓCZKAI K, et al. Authentication and authorization orchestrator for microservice-based software architectures[C]//Proceedings of 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics. Piscataway: IEEE Press, 2018: 1180-1184.

[6] BARDAS A G, SUNDARAMURTHY S C, OU X, et al. MTD CBITS: moving target defense for cloud-based IT systems[C]//Proceedings of 22nd European Symposium on Research in Computer Security. Berlin: Springer, 2017: 167-186.

[7] TORKURA K A, SUKMANA M I H, KAYEM A V D M, et al. A cyber risk based moving target defense mechanism for microservice architectures[C]//Proceedings of 2018 IEEE International Conference on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications. Piscataway: IEEE Press, 2018: 932-939.

[8] JIN H, LI Z, ZOU D Q, et al. DSEOM: a framework for dynamic security evaluation and optimization of MTD in container-based cloud[J]. IEEE Transactions on Dependable and Secure Computing, 2019, 18(3): 1125-1136.

[9] 张帅, 郭云飞, 孙鹏浩, 等. 云原生下基于深度强化学习的移动目标防御策略优化方案[J]. 电子与信息学报, 2023, 45(2): 608-616.
ZHANG S, GUO Y F, SUN P H, et al. Optimization scheme of moving target defense strategy based on deep reinforcement learning in cloud native environment[J]. Journal of Electronics & Information Technology, 2023, 45(2): 608-616.

[10] DUC T L, LEIVA R G, CASARI P, et al. Machine learning methods for reliable resource provisioning in edge-cloud computing: a survey[J]. ACM Computing Surveys, 2020, 52(5): 1-39.

[11] ZHOU D C, CHEN H C, SHANG K, et al. Cushion: a proactive resource provisioning method to mitigate SLO violations for containerized microservices[J]. IET Communications, 2022, 16: 2105-2122.

[12] YADAV T, RAO A M. Technical aspects of cyber kill chain[C]//Proceedings of International Symposium on Security in Computing and Communication. Berlin: Springer, 2015: 438-452.

[13] NOUREDDINE M A, FAWAZ A, SANDERS W H, et al. A game-theoretic approach to respond to attacker lateral movement[C]//Proceedings of 7th International Conference on Decision and Game Theory for Security. New York: ACM Press, 2016: 294-313.

[14] ALMOHRI H M J, WATSON L T, EVANS D. Misery digraphs: delaying intrusion attacks in obscure clouds[J]. IEEE Transactions on Information Forensics and Security, 2018, 13(6): 1361-1375.

[15] 张福, 程度, 胡俊. ATT&CK 框架实践指南[M]. 北京: 电子工业出版社, 2022.
ZHANG F, CHENG D, HU J. ATT&CK framework practice guide [M]. Beijing: Publish House of Electronics Industry, 2022.

[16] GAN Y, ZHANG Y Q, CHENG D L, et al. An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems[C]//Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM Press, 2019: 3-18.

[17] CHO J H, SHARMA D P, ALAVIZADEH H, et al. Toward proactive, adaptive defense: a survey on moving target defense[J]. IEEE Communications Surveys & Tutorials, 2020, 22(1): 709-745.

[18] CAI G L, WANG B S, HU W, et al. Moving target defense: state of the

art and characteristics[J]. Frontiers of Information Technology & Electronic Engineering, 2016, 17(11): 1122-1153.

[19] ZHANG X, SEN S, KURNIAWAN D, et al. E2E: embracing user heterogeneity to improve quality of experience on the web[C]//Proceedings of the ACM Special Interest Group on Data Communication. New York: ACM Press, 2019: 289-302.

[20] GIAS A U, CASALE G, WOODSIDE M. ATOM: model-driven autoscaling for microservices[C]//Proceedings of 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). Piscataway: IEEE Press, 2019: 1994-2004.

[21] LI Z, JIN H, ZOU D Q, et al. Exploring new opportunities to defeat low-rate DDoS attack in container-based cloud environment[J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 31(3): 695-706.

[22] BHASI V M, GUNASEKARAN J R, THINAKARAN P, et al. Kraken: adaptive container provisioning for deploying dynamic DAGs in serverless platforms[C]//Proceedings of the ACM Symposium on Cloud Computing. New York: ACM Press, 2021: 153-167.

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

[24] ABELS A, ROIJERS D, LENAERTS T, et al. Dynamic weights in multi-objective deep reinforcement learning[C]//Proceedings of International Conference on Machine Learning. Saarland: DBLP, 2019: 11-20.

[作者简介]

周大成（1995− ），男，河南息县人，信息工程大学博士生，主要研究方向为网络空间安全、云计算等。

陈鸿昶（1964− ），男，河南新密人，博士，信息工程大学教授、博士生导师，主要研究方向为网络空间安全、数据分析等。

何威振（1996− ），男，安徽亳州人，信息工程大学博士生，主要研究方向为网络空间安全、云计算等。

程国振（1986− ），男，山东菏泽人，博士，信息工程大学副教授、硕士生导师，主要研究方向为网络空间安全、软件定义网络等。

扈红超（1982− ），男，河南商丘人，博士，信息工程大学教授、博士生导师，主要研究方向为网络空间安全、拟态防御等。