计算机科学
COMPUTER SCIENCE

基于自适应遗传算法的微服务移动目标防御策略

刘轩宇, 张帅, 霍树民, 商珂

引用本文

刘轩宇, 张帅, 霍树民, 商珂. 基于自适应遗传算法的微服务移动目标防御策略[J]. 计算机科学, 2023, 50(9): 82-89.

LIU Xuanyu, ZHANG Shuai, HUO Shumin, SHANG Ke. Microservice Moving Target Defense Strategy Based on Adaptive Genetic Algorithm [J]. Computer Science, 2023, 50(9): 82-89.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.221000199

基于自适应遗传算法的微服务移动目标防御策略

刘轩宇　张　帅　霍树民　商　珂
战略支援部队信息工程大学信息技术研究所　郑州 450000
(13603696199@163.com)

摘　要　微服务架构因具有灵活、可扩展等特性,能够有效地提高软件的敏捷性,成为目前云中应用交付最主流的方法。然而,微服务化拆分使得应用的攻击面呈爆炸式增长,给以“要地防御”为核心的移动目标防御策略设计带来了巨大的挑战。针对该问题,提出了一种基于自适应遗传算法(AGA)的微服务移动目标防御策略,即动态轮换策略(DRS)。首先,基于微服务的特点,对攻击者的攻击路径进行分析;然后,提出微服务攻击图模型来形式化各种攻击场景,并对移动目标防御策略的安全增益和防御回报率进行定量分析;最后使用 AGA 求解移动目标防御的最优安全配置,即微服务的最优动态轮换周期。实验表明 DRS 具有可扩展性,相比统一配置策略、DSEOM 以及随机配置策略,其防御回报率分别提高了 17.25%,41.01%和 222.88%。

关键词:云计算;微服务;自适应遗传算法;移动目标防御
中图法分类号　TP393

Microservice Moving Target Defense Strategy Based on Adaptive Genetic Algorithm

LIU Xuanyu,ZHANG Shuai,HUO Shumin and SHANG Ke
Institute of Information Technology,PLA Strategic Support Force Information Engineering University,Zhengzhou 450000,China

Abstract　Microservice architecture can effectively improve the agility of software due to its flexible,scalable and other characteristics,and has become the most mainstream method of application delivery in the cloud. However,the microservice splitting makes the attack surface of applications grow explosively,which brings great challenges to the design of mobile target defense strategy with the core of “strategic defense”. To solve this problem,a microservice moving target defense strategy based on adaptive genetic algorithm(AGA),namely dynamic rotation strategy(DRS),is proposed. Firstly,based on the characteristics of microservice,the attack path of attackers is analyzed. Then,a microservice attack graph model is proposed to formalize various attack scena-rios,and the security gains and return of defense(RoD) of moving target defense strategies are quantitatively analyzed. Finally,AGA is used to solve the optimal security configuration of mobile target defense,that is,the optimal dynamic rotation cycle of microservices. Experiments show that DRS is scalable,and the defense return rate of DRS increases by 17.25%,41.01% and 222.88% respectively compared with the unified configuration strategy,DSEOM and random configuration strategy.

Keywords　Cloud computing,Microservice,Adaptive genetic algorithm,Moving target defense

1　引言

云计算是一种分布式计算模式,被广泛用于整合计算资源[1],使得全球网络连接大幅增加[2]。为了能够有效地利用云资源并加速开发和交付过程,微服务架构应运而生。在基于微服务的云环境中,传统的单体式应用按照逻辑功能被解耦为一组执行单一功能的独立微服务,并且每个微服务都允许被独立开发、部署、扩展和版本控制[3],多个微服务之间通常使用 HTTP 或 RPC 协议相互调用形成服务链,以此向用户提供集成功能。另外,每个微服务都会根据并发请求动态调整其副本数量[4],这大大提高了软件的敏捷性和灵活性[5]。由于微服务具有弹性、可扩展性、自动化、连续部署以及快速响应需求等特点,微服务体系架构在许多 IT 行业中得到了广泛应用,Amazon,Netflix,Twitter 和 Uber 等众多组织都使用微服务来交付它们的软件[6]。

各个微服务之间是松耦合的,因此微服务可以灵活更新。一天内微服务可能会更新几十甚至上百次[7],这样会导致云环境的复杂性大幅提高。同时,从单体式应用过渡到微服务架构,攻击面的数量激增,且微服务的攻击面不断发生变化,使得微服务攻击面难以管控[8]。由于当前网络空间攻防双方具有不对等的特性,攻击者可以通过嗅探或扫描技术探测系统,寻找可以被其利用且能够最大限度提高攻击效益的安全

到稿日期:2022-10-24　返修日期:2023-02-08
基金项目:国家自然科学基金(62072467);国家重点研发计划(2021YFB1006200,2021YFB1006201)
This work was supported by the National Natural Science Foundation of China(62072467) and National Key Research and Development Program of China(2021YFB1006200,2021YFB1006201).
通信作者:张帅(2012301200229@whu.edu.cn)

漏洞并对其发起攻击，进而引发一些安全问题，如数据泄露、拒绝服务、越权访问等[9]。

移动目标防御(Moving Target Defense，MTD)作为一种主动防御技术，其通过构建更难察觉和预测的动态系统以及限制漏洞的暴露时间来实现主动防御[10]，例如，改变或重组IP地址[11-12]、重排网络拓扑[13-14]、移动或迁移计算实例[15]等。MTD与传统防御技术最主要的不同是，传统防御技术被动响应攻击，而MTD主动转移攻击面[16]。

尽管MTD策略可以有效提高系统的安全性，但同时也会引入额外的开销，对服务质量造成影响。因此要考虑MTD策略对微服务的安全性与系统开销之间的平衡。与此同时，求解MTD策略配置本身就是一项困难的工作，加之应用在复杂的微服务场景中，使其变得更具挑战性，主要原因如下：

(1)基于微服务的云场景规模大，每个应用都包含多个微服务，并且每个微服务还有多个副本，因此微服务化会导致应用的攻击面数量骤增，为MTD策略设计带来巨大挑战。

(2)微服务场景下攻击面难以管控，攻击者到攻击目标的路径多样化，因此需要较高强度的MTD策略来对云中的重要路径进行保护。

针对以上问题，本文提出了一种基于自适应遗传算法(Adaptive Genetic Algorithm，AGA)的MTD策略，引入动态轮换策略(Dynamic Rotation Strategy，DRS)来保护云环境中的微服务，即在可接受的条件下，足够频繁地轮换微服务以避免攻击者的攻击。本文通过攻击图(Attack Graph，AG)模型形式化模拟微服务的复杂攻击场景，并利用AGA求解微服务的动态轮换周期，以实现防御回报率与系统开销之间的平衡。另外，对微服务的动态轮换周期进行建模，根据AG中边的权重(漏洞利用难度)计算出攻击者到攻击目标的最短路径，然后对该路径上的所有微服务执行DRS。

本文的主要贡献如下：

(1)提出了动态轮换策略，能够在一定程度上抵御攻击者对特定微服务的攻击，并利用AGA求解动态轮换周期。

(2)使用攻击图模型模拟微服务场景下的微服务拓扑结构，包括微服务之间的调用关系以及攻击者的攻击路径，并使用Dijkstra算法在攻击图中计算攻击者到攻击目标的最短路径。

(3)通过实验验证了所提方法的有效性和可扩展性，并与统一配置策略、DSEOM以及随机配置策略进行比较，防御回报率分别提高了17.25%，41.01%，222.88%。

2 相关工作

近年来，受新冠肺炎的影响，云解决方案在教育、医疗、电子商务等众多行业中的应用越来越广泛[17]。云计算不仅可以降低资源共享的成本，还可以随时为世界各地的用户不间断地提供服务，但同时也引发了许多安全威胁。

目前，业内对微服务架构的防护问题进行了广泛的研究，已有技术手段大致可以分为两大类，一类是传统防御技术，另一类是主动防御技术。传统的防护策略主要使用防火墙[18]、入侵检测等技术抵御攻击，然而这些静态的防御技术不足以应对微服务架构中的安全问题。主动防御技术则通过引入动态性、异构性和冗余性等机制，可从参数、软件、平台等多个层面采取相应策略来实施防御，代表性的技术有MTD、拟态防御等。同时，这些机制的引入会对系统开销和服务连续性造成一定的影响。因此，恰当分析建模威胁和工作场景，选取恰当的防御策略并进行评估，是主动防御发挥最佳效果的关键。

Jin等[16]提出了一种动态安全评估与优化的MTD策略，其引入中介中心性的概念来评估云环境中节点的重要性，并对关键节点实施MTD策略。虽然该策略能够有效地提高云环境的安全性，但文中并未给出关键节点的评判标准，这样可能会增加系统开销且达不到最优的防御效果。Zhang等[8]对文献[16]中提出的“要地”保护策略做出了改进，提出了一个基于DQN算法的MTD策略优化方案，该方案对云中所有的微服务都使用动态清洗的主动防御策略，通过调整微服务的动态清洗周期来优化整体的防御效率，并且该策略无需防御者调节参数就可以实现安全与开销的折中。Bardas等[19]提出了一个可以捕获云环境中服务依赖性的MTD平台，通过实时替换计算实例的最佳策略来最大限度地增大攻击者的攻击难度。Zeng等[20]提出了一种动态异构调度方法，该方法基于MTD思想对动态异构式系统中的计算实例执行实时动态调度，可以在不影响正常服务的前提下阻断攻击者的攻击行为。Connell等[21]分析了重新配置资源所需的时间及重新配置率与攻击者的成功概率之间的关系，并提出了一个用于评估MTD对重新配置资源的可用性和性能影响的定量分析模型。该模型允许防御者根据重新配置时间选择重新配置率和重新配置技术。Maleki等[22]提出了一个基于马尔可夫模型的MTD分析框架，该框架允许对广泛的MTD策略进行建模，并给出了攻击者的成功概率与所花费的时间/成本之间的关系，同时还展示了如何使用他们的方法分析MTD的组成。Peng等[23]提出了一种基于概率部署的MTD策略，该策略考虑了攻击者对目标服务攻击面的累积影响，并基于云服务攻击面与攻击时间的关系，利用云服务的异构性与动态性使服务更具抵御攻击的能力。实验证明了所提出的MTD策略在使用异构和动态攻击面保护云服务方面的有效性。Hong等[24]提出使用分层攻击表示模型(Hierarchical Attack Representation Model，HARM)来分析和评估不同MTD策略的有效性和可扩展性，并使用重要性度量(Importance Measures，IM)来计算重要节点，同时在这些节点上部署MTD策略。Alavizadeh等[11,25]提出了一种使用安全建模形式化来评估组合MTD技术有效性的方法，该方法使用网络中心度量(Network Centrality Measures，NCM)计算云中的关键组件，并分别从系统风险和可靠性的角度评估其安全性。

3 模型建立

本节首先介绍云环境下的威胁模型，然后利用攻击图模型建模基于微服务的云攻击场景，并结合DRS分析其安全性。

3.1 威胁模型

假设云平台和服务提供商是可信的且攻击者来自外部网络，攻击者的目的是获取系统上未经授权的权限或窃取用户

的隐私数据。图1给出了一个基于微服务环境的攻击场景示例，对攻击目标、攻击过程及攻击能力的描述如下。

(1)攻击目标：云环境下所有运行的微服务都有可能成为攻击者的攻击目标。微服务是由服务代码及其依赖库组成的服务应用程序，本文假设其中有一些可以被攻击者利用的漏洞。

(2)攻击过程：假设攻击者根据网络杀伤链理论[26]发起攻击。具体来说，攻击者首先执行各种侦察探测以识别攻击目标中的漏洞，当攻击者掌握了一个或多个攻击目标中的可利用漏洞，并且假设攻击者具有利用一个或多个攻击成功的已知漏洞的经验，针对这些漏洞选择攻击方法(如执行恶意代码或横向移动等)，破坏攻击目标，实现对攻击目标的持续利用。

(3)攻击能力：本文只关注应用程序自身的漏洞以及微服务之间的横向移动攻击。假设攻击者来自外部网络，并且只可以向对外开放端口的公共服务发起攻击。若图1中的服务 A 被攻击者成功利用后，攻击者可以继续对与服务 A 存在依赖关系的服务发起攻击，即服务 C，以实现在服务间的横向移动。假设云环境中网络配置是根据网络隔离策略制定的，只有当微服务之间有依赖关系时，两个微服务间的网络才是可达的。

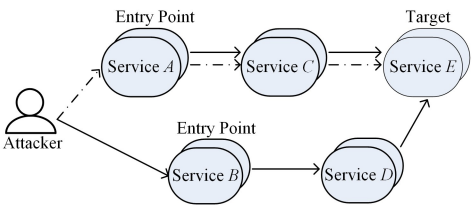


图1 基于微服务环境的攻击场景示例

Fig. 1 Example of attack scenario based on microservice environment

此外，攻击者的攻击能力也是受限制的，假设攻击者不知道微服务的配置细节，包括微服务的编排、微服务在云中的放置以及微服务的副本数量。需要强调的是，本文假设攻击者的目标只有运行中的任务，而动态轮换过程不受攻击的影响。基于上述分析，解决云环境下微服务的安全问题是具有挑战性的。

3.2 DRS 原理

基于上述威胁模型，本文结合 MTD 思想提出一种静态防御策略 DRS，通过周期性轮换攻击者到攻击目标最短路径上所有微服务的方法，实现阻断攻击者渗透云环境获取重要数据或横向移动的攻击过程。假设防御者已知云中微服务的所有配置信息，包括副本数量、微服务之间的调用关系以及各项微服务的漏洞信息等，并且 DRS 在部署云中微服务之前就已经被提前求解。总体来说，首先根据微服务之间的拓扑关系，利用 Dijkstra 算法确定攻击者到攻击目标的最短路径，然后使用 AGA 求解其上所有微服务的动态轮换周期，并在云中对相应微服务进行策略部署，最后在云中创建具有 DRS 的微服务。

上述策略能够抵御一定程度的攻击。具体而言，若微服务的动态轮换周期小于攻击者所需的攻击时间，那么攻击者就无法完成攻击；同样地，若攻击者已经攻破了某个微服务，并试图发起横向移动攻击，那么引入 DRS，定期更换云环境中的微服务，使被挟持的微服务恢复清洁状态，就可以使攻击者失去对微服务的控制。

3.3 理论模型

AG 是一种常用于分析网络脆弱性的技术[27]，其涵盖了攻防场景中的所有攻击事件。根据上述威胁模型，对 AG 做出如下定义。

定义 1 将 AG 表示为一个有向无环图 $G=(N,E)$。N 为图中所有节点的集合，在微服务场景下可以形式化为全部的攻击目标，即 $N=\{n_s \mid s\in S\}$，其中 S 表示所有微服务副本集合；$E\subseteq N\times N$ 是图中所有边的集合，每条边都表示有序节点对 (N_a,N_b) 之间的攻击路径，其中 $N_a,N_b\in N$ 且 $a\neq b$。攻击路径表示在攻击者成功控制前驱节点的条件下，利用后继节点的漏洞发起攻击的行为。

定义 2 将节点被成功攻击的概率定义为 AG 中边的权重，该值与两个指标有关：节点中漏洞的利用难度 ED 以及节点的动态轮换周期 T。

漏洞利用难度表示通过至少一个漏洞破坏攻击目标的难度。为了评估节点漏洞利用难度，本文使用通用漏洞评分系统 CVSSv3.1[28]中定义的基础分数指标(Base Score Metrics)中的可利用性指标(Exploitability Metrics，EM)和时间指标(Temporal Metrics)中的利用代码成熟度(Exploit Code Maturity，ECM)来量化节点中漏洞的利用难度[29]，其中 EM 量化每个漏洞的可利用性，ECM 计算漏洞被攻击的可能性。EM 的计算式如式(1)所示：

$$EM=(8.22\times AV\times AC\times PR\times UI)^{-1} \tag{1}$$

其中，AV(Attack Vector)代表攻击向量，该指标反映了可能利用漏洞的环境；AC(Attack Complexity)代表攻击复杂度，该指标描述了为了利用漏洞而必须存在的超出攻击者控制范围的条件；PR(Privileges Required)代表权限要求，该指标描述了攻击者在成功攻击脆弱组件前必须拥有的权限级别；UI(User Interaction)代表用户交互，该指标描述了攻击脆弱组件对除攻击者之外的用户参与的需求。

另外，由于攻击者能力及偏好未知且每个攻击目标中都有多个可以被利用的漏洞，因此每个漏洞都有被攻击的可能性。将 ECM 作为 EM 的权重计算节点漏洞利用难度 ED。

$$ED=\frac{\sum\limits_{V}(ECM\times EM)}{\sum\limits_{V}ECM} \tag{2}$$

其中，V 表示节点中所有漏洞的集合。

为了研究微服务被成功攻击的概率与动态轮换周期之间的关系，本文假设攻击者在时间充足的情况下能够获取攻击所需的信息并渗透系统获取未经授权的权限或窃取重要数据，因此动态轮换周期越长，攻击者成功的可能性就越大。本文使用指数函数来表示每个独立的微服务被攻击者成功破坏的概率与攻击时间的关系[21]。具体公式如下：

$$P_A(t)=1-\frac{1-e^{(t-T_s)}}{1-e^{-T_s}} \tag{3}$$

其中，T_s 表示攻击者达到最大成功概率所需的时间，该指标可以间接反映漏洞利用难度[8]，因此使用 $T_s=\mathcal{F}(ED)$ 表示二者

的映射关系。在函数的初始阶段攻击者以低速率收集漏洞信息，随着时间的推移，所积累的漏洞信息量以指数形式增加，致使攻击成功概率不断变大，并在T_s时刻达到最大值。

定义 3 在给定的 AG 中，攻击者N_{att}到攻击目标N_{tar}之间的最短路径为 $AP: N_{att} \rightarrow N_{tar}$，攻击难度可以表示为 $ED(att,tar)$。AP 上的所有节点即为本文要部署 DRS 的节点。

3.4 安全分析

假设微服务环境中有 R 个微服务 $M=\{M_1, M_2, \cdots, M_R\}$，并且每个微服务都有相应的轮换周期 $T=\{T_1, T_2, \cdots, T_R\}$。在确定每个微服务的副本数后，就可以根据 3.2 节所述部署 AG。

基于定义 3，对于部署 DRS 的节点，假设每次执行轮换动作后攻击者都要重新启动攻击流程，这就导致攻击者需要付出更大的代价来达到攻击目的。式(4)表示执行动态轮换后节点漏洞的利用难度，该公式也可以形式化地表示部署 DRS 后的攻击成本。

$$ED' = \sum_{V} \frac{ED}{T_i} \tag{4}$$

其中，V 表示节点中所有漏洞的集合，T_i 为节点的动态轮转次数。

由于无法判断攻击者的攻击目标，根据上述对攻击者能力的描述，假设攻击者会选择攻击成功率最大的路径对目标展开攻击，因此本文采用攻击者到攻击目标的最短路径作为评估 DRS 有效性的方法，同时本文也根据上述最短路径来确定部署 DRS 的节点。相应地，DRS 所带来的安全增益也可以通过最短路径的攻击难度的增量 $\Delta ED(att,tar)$ 来量化，即：

$$\Delta ED(att,tar) = \sum_{AP} [ED' - ED] \tag{5}$$

对于动态轮换周期，若设置太短，则会浪费云资源并影响微服务的服务质量(Quality of Service，QoS)，若设置太长，微服务会频繁地受到攻击者的攻击。因此本文采用微服务的动态轮换频率作为 DRS 的开销指标 $\mathscr{C}$，计算式如下：

$$\mathscr{C} = \sum_{AP} \frac{1}{T_i} \tag{6}$$

为了最大化 DRS 的防御效率，本文将防御回报率(Return of Defense，RoD)定义为安全增益与系统开销的比值。同时通过最大化 RoD 求解最优动态轮换周期，该问题可表示为：

$$\max \frac{\Delta ED(att,tar)}{\mathscr{C}} \quad \text{s.t. } T_i \in T, T_{\min} \leqslant T_i \leqslant T_{\max} \tag{7}$$

4 基于 AGA 的安全配置算法

AGA 是一种基于生物进化理论和自然遗传机制演变的元启发式随机搜索算法，被广泛用于解决有约束或无约束的优化和安全问题[30-31]。其不同于传统 GA，传统 GA 的交叉概率和变异概率都会设置为固定值，因此不论个体是否优良，都会以相同概率进行交叉和变异，这样就会导致一些问题的出现。

(1)影响算法效率。对于优良个体，应该尽量减小交叉和变异概率，使其尽可能被保留下来；而对于劣质个体，应增大其交叉和变异概率，使其在进化过程中被淘汰。

(2)不能很好地满足种群进化过程中的需求。实际上，在进化初期，种群需要较大的交叉和变异概率，使得算法能够快速找到最优解的范围并且避免陷入局部最优；而在进化后期，种群需要较小的交叉和变异概率，使算法在找到全局最优解后能够快速收敛，缩短算法执行时间，提高算法效率。

AGA 针对上述问题对传统 GA 进行了改进，其可以在进化过程中根据适应度函数的值自适应调整交叉概率与变异概率，并在完整搜索空间中快速找到全局最优解。AGA 旨在达到种群多样性与收敛性之间的平衡，防止 AG 陷入局部最优解。

本文采用 AGA 解决式(7)所示的问题，当适应度陷入局部最优，即个体分布较为集中，种群多样性较低时，交叉概率和变异概率会相应增大；相反，当适应度离散，即个体分布较为分散，种群多样性较高时，交叉和变异概率会自动减小。下面将针对 AGA 中的一些概念和遗传算子进行详细介绍。

4.1 AGA 中的一些概念

种群是搜索空间中的一组可能解，即根据适应度函数选择出来的一组解；个体是种群中的每一个可行解；染色体是对每个可行解的编码，可以使用二进制编码、十进制编码或十六进制编码技术；基因表示染色体中的每一位编码，即每个可行解中的分量；适应度函数用来评估每个个体在种群中的适配程度。

在本文中，种群为一组可能的微服务安全配置，个体为种群中对应的每条最短路径上微服务的安全配置，染色体表示对每条最短路径的安全配置进行编码。由于本文使用十进制编码，因此该值即为需要执行动态轮换的微服务的轮换周期的集合。基因表示最短路径上每个微服务的具体安全配置，适应度函数 $f(\cdot)$ 就是评估函数，即式(5)。

4.2 遗传算子

遗传算子可以分为选择算子、交叉算子和变异算子。下面将分别对这 3 种算子进行介绍。

4.2.1 选择算子

选择算子是 AGA 中每个循环的开始，目的是根据概率从当前种群中选择个体，从而产生新一代种群。本文使用的选择技术是轮盘赌选择，其中个体被选中的概率与其适应度大小成正比，公式如下：

$$P_s = \frac{f(x_i)}{\sum_{x_i \in X} f(x_i)} \tag{8}$$

其中，$f(x_i)$ 是个体 x_i 的适应度，$\sum_{x_i \in X} f(x_i)$ 表示种群的适应度。根据 P_s 画出选择轮盘，固定选择指针，转动轮盘随机选择个体，由于优质个体的适应度高，在轮盘中所占比例大，因此被选中的可能性很大。该方法可以大概率地将优质个体保留到下一代种群中，有利于 AGA 找到全局最优解。

4.2.2 自适应交叉算子

自适应交叉算子是 AGA 中的一个重要步骤，为了增加种群的多样性，提高 AGA 的全局搜索能力，需要在种群中随机选择两个个体进行交叉操作，具体公式为：

$$P_c = \begin{cases} k_1 \dfrac{f_{\max} - f'}{f_{\max} - f_{\text{avg}}}, & f' > f_{\text{avg}} \\ k_2, & f' \leqslant f_{\text{avg}} \end{cases} \tag{9}$$

其中，个体以概率 P_c 进行交叉操作，$f_{\max}$ 为当前种群中的最大

适应度，$f'=\max(f_a,f_b)$表示进行交叉操作的两个个体适应度的最大值，f_{avg}为种群的平均适应度，参数k_1，k_2均为[0,1]间的常数。因此，对高适应度的个体实施低交叉率，有利于保留优质基因；对适应度低的个体实施高交叉率，有利于淘汰劣质基因。

4.2.3 自适应变异算子

自适应变异算子在 AGA 中与自适应交叉算子同样重要，其能够提高 AGA 的局部搜索能力。变异算子是针对某一个体，以一定的概率改变其分量，公式如下：

$$P_m=\begin{cases}k_3\dfrac{f_{max}-f}{f_{max}-f_{avg}}, & f>f_{avg}\\ k_4, & f\leqslant f_{avg}\end{cases} \tag{10}$$

其中，个体中的分量以P_m进行变异，f 表示进行变异操作个体的适应度，f_{max}与f_{avg}表示的含义与式(9)中相同，参数k_3，k_4均为[0,1]间的常数。因此，对高适应度的个体实施低变异率，有利于保留优质基因；对适应度低的个体实施高变异率，有利于淘汰劣质基因。

AGA 中的自适应交叉算子与自适应变异算子在个体层面和种群层面都能进行较好的自适应，但是对于种群中适应度最高的个体，其自适应交叉算子与自适应变异算子的值均为 0，这就使得该个体会被保留到下一代种群中。结合选择算子，可能会导致该个体以指数形式在新种群中增长，这会引起 AGA 过早收敛或陷入局部最优，因此 AGA 为每个个体都引入一个突变概率P_m'，本文将P_m'设置为 0.005。AGA 算法的具体流程如算法 1 所示。

算法 1 AGA 的运行流程

输入：种群规模 popsize，迭代次数 NGEN

输出：微服务的最优安全配置

1. 随机生成微服务安全配置范围 $\mathcal{R}$
2. pop←$\mathcal{R}$，popsize / * pop 为初始化集群 * /；
3. ind_{best}←selectBest(pop) / * 从 pop 中选择适应度最大的个体 * /；
4. for episode in range(NGEN)：
5. selectpop=Selection(pop,popsize) / * selectpop 为新一代种群 * /；
6. while len(nextoff)！=popsize / * nextoff 为下一代种群 * /；
7. offs=[selectpop.pop()for_inrange(2)] / * 将 selectpop 根据适应度升序排列，并删除适应度最小的两个个体，生成子代种群 * /；
8. r=random.random()；
9. if r<P_c，then
10. i_1，i_2=Crossoperate(offs) / * 从 offs 中随机选择两个个体进行交叉 * /；
11. if r<P_m，then
12. i_3=Mutation(i_1)；
13. i_4=Mutation(i_2) / * 对交叉个体进行变异 * /；
14. nextoff←i_3，i_4 / * 将变异后的个体加入下一代种群 * /；
15. else
16. nextoff←i_1，i_2 / * 将交叉后的个体加入下一代种群 * /；
17. else
18. nextoff←offs / * 将子代加入下一代中 * /；
19. pop=nextoff / * 更新当前种群 * /；
20. best_ind=max(ind_{best}，selectbest(pop) / * 选出适应度最高的个体 * /；
21. END
22. 获取最优个体，即微服务的最优安全配置。

AGA 最初会随机生成一个初始种群，即父代；计算适应度，然后经过选择、交叉、变异等遗传算子生成新种群，即子代，重复进化过程，直至选择出最合适的基因，并组合成最优秀的个体。

5 实验与分析

本章在实际环境下验证微服务动态轮换方法的有效性。

5.1 实验环境与参数设置

本文实验采用云编排平台 Kubernetes 搭建微服务云环境，包括 7 台服务器，配置均为 32 核，1.50 GHz，32 GB 内存，其中 1 台为控制节点，6 台为计算节点。在上述云平台上部署了一个由 5 项微服务组成的 web 应用，微服务之间的关系如图 1 所示，其漏洞的各项信息如表 1 所列。需要说明的是，表 1 中 ECM 的值是从 Exploit Database 中获取的最新值，其上限为 10，数值越大表示被利用的可能性越大。本文参照文献[21]中对攻击者能力的假设，根据式(3)可知，在给定微服务攻击难度的情况下，微服务被成功攻击的概率会根据轮换周期的不同而发生改变。

表 1 微服务应用漏洞

Table 1 Microservice application vulnerabilities

Microservice	Attack×Target	CVE ID	EM	ECM	ED
A	Apache	CVE-2022-39135	3.9	9.2	2.88
		CVE-2022-34305	2.8	5.6	
		CVE-2019-10095	3.9	8.9	
		CVE-2020-13958	1.8	7.1	
		CVE-2018-11776	2.2	7.9	
		CVE-2021-44521	2.3	8.3	
B	Memcached	CVE-2016-8706	2.2	6.8	3.41
		CVE-2016-8705	3.9	8.3	
		CVE-2016-8704	3.9	8.3	
C	Tomcat	CVE-2022-25762	3.9	7.8	3.40
		CVE-2022-34305	2.8	5.6	
		CVE-2019-10104	3.9	8.6	
		CVE-2020-1938	3.9	8.9	
		CVE-2019-0232	2.2	7.9	
D	ImageMagick	CVE-2017-14650	2.2	6.8	2.97
		CVE-2017-14224	2.8	8.3	
		CVE-2021-20311	3.9	7.1	
E	Mysql	CVE-2016-6662	3.9	8.8	3.33
		CVE-2020-14878	2.1	7.3	
		CVE-2018-2696	3.9	6.8	

根据微服务应用类型，本文假设攻击者的攻击目标为 Mysql，目的是窃取 Mysql 中的重要数据。在生成 AG 时，对于无状态应用 Apache，Memcached，Tomcat 以及 ImageMagick，其副本能够单独提供服务；而对于有状态应用 Mysql，其通过集群的方式对外提供服务。因此本文考虑了在 AG 中以 1 个节点代替多个 Mysql 副本构成的 Mysql 集群。

在算法层面，将微服务的配置信息以数组的形式输入 AGA 中，其中数组维度代表微服务的数量，数组中的数据代表每项微服务的副本数，代码执行后输出每个微服务的最优动态轮换周期。为了模拟实际生产中的复杂环境，本文考虑

了微服务规模较大的场景，通过改变 AGA 的输入，增加输入数据的维度即可模拟大规模微服务场景。同时，在其他关于微服务场景的一些研究中，如文献[8,16,29]等也使用少量微服务拓扑模拟复杂场景。然而，增加 AGA 的输入数据量会导致代码执行时间和复杂度增加，在 5.2 节会进行具体分析。

为了防止 AGA 陷入局部最优解，本文设定参数$k_1=k_2=1$，这使得适应度不超过种群平均适应度的个体必须进行交叉操作，从而提高 AGA 的全局搜索能力；同理，设定参数$k_3=k_4=0.5$，利用适应度低于种群平均适应度的个体在搜索空间中寻找包含最优解的范围，因为这种个体本就应该被打乱。本文使用 Dijkstra 算法[2]计算最短路径，并对最短路径上的微服务进行动态轮换。

5.2 实验分析

为了验证 DRS 的性能和可扩展性，本文首先在实验场景下部署不同副本数量的微服务应用，然后在不改变微服务副本数量的前提下，逐渐增加 AGA 的迭代次数，并分别对防御回报率和时间开销进行量化和测量。

图 2 给出了 DRS 的防御回报率随微服务副本数和 AGA 迭代次数的变化趋势。从整体上看，DRS 所产生的防御回报率随微服务副本数量的变化趋势大致相同，并且收敛后的防御回报率相等。如图 2(a)－图 2(d)所示，在微服务副本数分别为 100，200，300 和 400 的情况下，DRS 的防御回报率最后都收敛于 38.84。对于图 2 中的每一幅子图，在微服务副本数量固定的条件下，对 AGA 分别进行了 50，100，200 和 500 次的迭代。实验结果表明：(1)微服务副本数量越大，AGA 收敛到最优解的迭代次数就越多，例如，对比图 2(a)与图 2(b)可知，当微服务副本数量为 100 时，AGA 分别在 34 次、43 次、121 次和 160 次时收敛，而当微服务副本为 200 时，AGA 分别在 31 次、81 次、151 次和 216 次时收敛；(2)AGA 的迭代次数越多，所得到的防御回报率就越大。如图 2(c)所示，当微服务副本数为 300 时，AGA 在不同迭代次数下防御回报率的收敛值分别为 36.62，38.05，38.84 以及 38.84。综合来看，AGA 在收敛时间上表现良好，防御回报率在进化后期趋于稳定，基本都在迭代 280 次左右开始收敛。

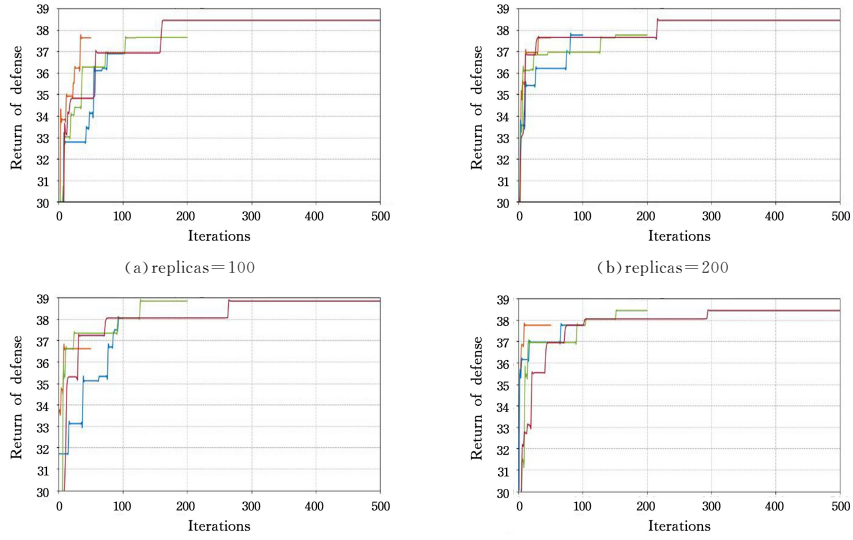


图 2 DRS 的防御回报率随微服务副本数和 AGA 迭代次数的变化

Fig. 2 Security gain of DRS varies with the number of microservice replicas and AGA iterations

图 3 给出了在不同微服务副本数量和迭代次数的条件下执行 AGA 的时间开销。

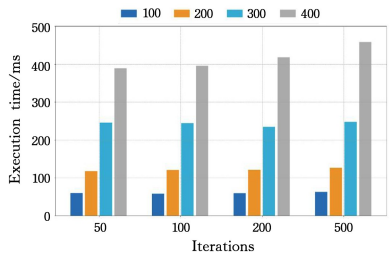


图 3 AGA 在不同微服务副本数与迭代次数下的执行时间

Fig. 3 Execution time of AGA with different number of microservice replicas and iterations

根据图 3 中的数据可以得出：(1)微服务副本数量固定时，迭代次数越多，AGA 的执行时间就越长，例如，当微服务副本数量为 400 时，分别迭代 50 次、100 次、200 次和 500 次所需的时间为 62.62 ms，126.25 ms，247.5 ms 和 417.76 ms；(2)当 AGA 的迭代次数固定时，微服务副本数量越大，则时间开销越大，例如，AGA 迭代次数为 200 时，微服务副本数分别为 100，200，300，400 的时间开销分别为 58.03 ms，120.47 ms，244.08 ms 以及 359.43 ms。这是因为 AGA 的种群变大，搜索范围扩大，求得最优解的能力下降，因此需要的时间就越长。

综合图 2、图 3 可得，微服务副本数量和迭代次数增多，都会导致 AGA 的时间开销增加，但收敛结果更优。因此，选取合适的微服务副本数和迭代次数，才能最大化 AGA 的性能。

5.3 对比实验分析

本文将 DRS 与其他方法进行了对比实验，图 4 给出了 AGA 与 DSEOM、统一配置策略以及随机配置策略在相同微服务规模下所产生的防御回报率的对比结果。其中，AGA 的

防御效率选取微服务副本数量300、迭代500次所得的防御回报率。下面对各策略进行详细说明。

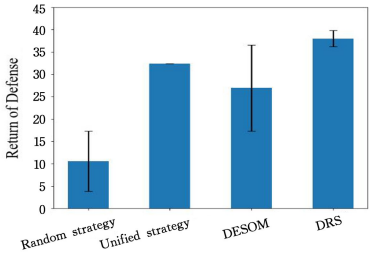


图4 不同策略的防御回报率对比

Fig. 4 Comparison of security gains of different strategies

DSEOM[16]与本文相同，都使用攻击图模型描述云中微服务的攻击难度，并且都应用了最短路径来求取攻击图中的重要节点。但与本文不同的是，DSEOM的攻击场景中包含了服务层、虚拟化层和物理计算节点，而本文只考虑了基于微服务的云场景；DSEOM使用D-KIN算法计算攻击图中的关键节点，并对其实施安全防护策略，而本文使用Dijkstra算法计算经过目标节点的最短路径，然后对该路径上的所有节点都进行动态轮换。统一配置策略[19]简化了微服务的安全配置问题，为所有的微服务副本设置相同的动态轮换周期，降低了求解问题的复杂度。随机配置策略遍历云中所有的微服务副本，并为每一个微服务随机生成安全配置。

根据图4可知，随机配置策略的防御回报率最小，DSEOM的防御回报率标准差最大。造成该现象的原因为：(1)随机配置策略增强了安全配置的随机性与未知性，该策略下微服务的动态轮换周期可能会中断攻击者的攻击过程，也可能为攻击者达到攻击目的而提供便利；(2)DSEOM只为云环境中的重要节点提供安全防护，因此只要攻击者识别出这种防御手段，利用其他节点对目标节点进行攻击，该方法就会失去防护作用。另外，DSEOM中的参数需防御者自行配置，因此参数的选取也会导致防御回报率的变化。统一配置策略简化了安全配置问题，并通过遍历的方法选取简化后的最优配置，因此该策略的稳定性最好。AGA在经过500次迭代后产生的平均防御回报率最大，并且与DSEOM和随机配置策略相比，其防御回报率的标准差最小，基本可以实现稳定的防御效果。

结束语 本文提出了一个基于自适应遗传算法的微服务移动目标防御策略。首先，使用攻击图模型对云中微服务的复杂攻击场景进行建模，同时分析了微服务被成功攻击的概率与动态轮换周期之间的关系。然后，定量分析DRS的有效性，以最大化防御回报率求解最优防御配置的问题。最后，采用AGA对上述问题进行求解。实验证明了DRS的有效性及可扩展性。DRS可以从3个方面提高安全性。首先，DRS可以减少有效攻击的次数，根据网络杀伤链，一次完整的攻击包括7个步骤，假设攻击者只针对特定的微服务，并且攻击者发起一次完整的攻击需要时间a，如果被攻击的微服务在时间b($b<a$)内动态轮换一次，那么攻击者就无法完成一次完整的攻击；其次，DRS可以保持微服务的清洁状态，所提出的微服务轮换策略可以被看作是一种清理机制，一旦经过确定的轮换周期，DRS将按照优先级清理特定的微服务；最后，DRS可以防止攻击者不断泄露工作流数据，经过轮换周期后，DRS将回收指定的微服务，因此攻击者无法长时间窃取用户的隐私及敏感数据。

为了推进本文工作，提出以下未来研究方向：

(1)异构化微服务副本。在本文中，对于相同的应用，微服务副本都是同构的，因此对于攻击者来说，尽管微服务副本数量很大，其攻击难度与攻击一个应用的难度是相同的。

(2)考虑攻击者绕过最短路径上的节点对攻击目标实施攻击的情况。本文假设攻击者只会选择攻击成功概率最大的路径对攻击目标展开攻击，而在实际中，攻击者的行为不会这么理想化。

(3)考虑云中动态变化的复杂场景。DRS为静态策略，其先验地计算出应用的最短路径，并为其上的微服务计算动态轮换周期，具有一定的局限性。

参 考 文 献

[1] GAO X, STEENKAMER B, GU Z, et al. A study on the security implications of information leakages in container clouds[J]. IEEE Transactions on Dependable and Secure Computing, 2018, 18(1): 174-191.

[2] BUZACHIS A, CELESTI A, GALLETTA A, et al. Evaluating an application aware distributed Dijkstra shortest path algorithm in hybrid cloud/edge environments[J]. IEEE Transactions on Sustainable Computing, 2021, 7(2): 289-298.

[3] CERNY T, DONAHOO M J, PECHANEC J. Disambiguation and comparison of soa, microservices and self-contained systems[C]// Proceedings of the International Conference on Research in Adaptive and Convergent Systems. 2017: 228-235.

[4] PRACHITMUTITA I, AITTINONMONGKOL W, POJJANASUKSAKUL N, et al. Auto-scaling microservices on IaaS under SLA with cost-effective framework[C]// 2018 Tenth International Conference on Advanced Computational Intelligence(ICACI). IEEE, 2018: 583-588.

[5] SULTAN S, AHMAD I, DIMITRIOU T. Container security: Issues, challenges, and the road ahead[J]. IEEE Access, 2019, 7: 52976-52996.

[6] SOLDANI J, TAMBURRI D A, VAN DEN HEUVEL W J. The pains and gains of microservices: A systematic grey literature review[J]. Journal of Systems and Software, 2018, 146: 215-232.

[7] HEORHIADI V, RAJAGOPALAN S, JAMJOOM H, et al. Gremlin: Systematic resilience testing of microservices[C]// 2016 IEEE 36th International Conference on Distributed Computing Systems(ICDCS). IEEE, 2016: 57-66.

[8] ZHANG S, GUO Y F, SUN P H, et al. Deep Reinforcement Learning based Moving Target Defense Strategy Optimization Scheme for Cloud Native Environment[J]. Journal of Electronics & Information Technology, 2022, 44: 1-9.

[9] WANG Y, GUO Y, GUO Z, et al. Securing the intermediate data of scientific workflows in clouds with ACISO[J]. IEEE Access, 2019, 7: 126603-126617.

[10] ALAVIZADEH H, HONG J B, JANG-JACCARD J, et al. Com-

prehensive security assessment of combined MTD techniques for the cloud[C]// Proceedings of the 5th ACM Workshop on Moving Target Defense. 2018:11-20.

[11] ALAVIZADEH H, HONG J B, KIM D S, et al. Evaluating the effectiveness of shuffle and redundancy mtd techniques in the cloud[J]. Computers & Security, 2021, 102:102091.

[12] ALAVIZADEH H, JANG-JACCARD J, KIM D S. Evaluation for combination of shuffle and diversity on moving target defense strategy for cloud computing[C]// 2018 17th IEEE International Conference on Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference on Big Data Science And Engineering (TrustCom/BigDataSE). IEEE, 2018:573-578.

[13] 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.

[14] WANG L, WU D. Moving target defense against network reconnaissance with software defined networking[C]// International Conference on Information Security. Cham: Springer, 2016: 203-217.

[15] TORQUATO M, MACIEL P, VIEIRA M. Analysis of vm migration scheduling as moving target defense against insider attacks[C]// Proceedings of the 36th Annual ACM Symposium on Applied Computing. 2021:194-202.

[16] JIN H, LI Z, ZOU D, 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.

[17] YING F, ZHAO S, DENG H. Microservice Security Framework for IoT by Mimic Defense Mechanism[J]. Sensors, 2022, 22(6): 2418.

[18] 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.

[19] BARDAS A G, SUNDARAMURTHY S C, OU X, et al. MTD CBITS: Moving target defense for cloud-based IT systems[C]// European Symposium on Research in Computer Security. Cham: Springer, 2017:167-186.

[20] ZENG W, HU H C, LI L S, et al. Dynamic heterogeneous scheduling method based on Stackelberg game model in container cloud[J]. Chinese Journal of Network and Information Security, 2021, 7(3):95-104.

[21] CONNELL W, MENASCE D A, ALBANESE M. Performance modeling of moving target defenses with reconfiguration limits [J]. IEEE Transactions on Dependable and Secure Computing, 2018, 18(1):205-219.

[22] MALEKI H, VALIZADEH S, KOCH W, et al. Markov modeling of moving target defense games[C]// Proceedings of the 2016 ACM Workshop on Moving Target Defense. 2016:81-92.

[23] PENG W, LI F, HUANG C T, et al. A moving-target defense strategy for cloud-based services with heterogeneous and dynamic attack surfaces[C]// 2014 IEEE International Conference on Communications (ICC). IEEE, 2014:804-809.

[24] HONG J B, KIM D S. Assessing the effectiveness of moving target defenses using security models[J]. IEEE Transactions on Dependable and Secure Computing, 2015, 13(2):163-177.

[25] ALAVIZADEH H, KIM D S, JANG-JACCARD J. Model-based evaluation of combinations of shuffle and diversity MTD techniques on the cloud[J]. Future Generation Computer Systems, 2020, 111:507-522.

[26] HUTCHINS E M, CLOPPERT M J, AMIN R M. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains[J]. Leading Issues in Information Warfare & Security Research, 2011, 1(1):80.

[27] INGOLS K, LIPPMANN R, PIWOWARSKI K. Practical attack graph generation for network defense[C]// 2006 22nd Annual Computer Security Applications Conference (ACSAC'06). IEEE, 2006:121-130.

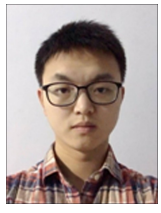
[28] FIRST. Common Vulnerability Scoring System v3.1: Specification Document[EB/OL]. https://www.first.org/cvss/v3.1/specification-document.

[29] LI H, GUO Y, SUN P, et al. An optimal defensive deception framework for the container-based cloud with deep reinforcement learning[J]. IET Information Security, 2022, 16(3): 178-192.

[30] CASAS I, TAHERI J, RANJAN R, et al. GA-ETI: an enhanced genetic algorithm for the scheduling of scientific workflows in cloud environments[J]. Journal of Computational Science, 2018, 26:318-331.

[31] TAHIR M, SARDARAZ M, MEHMOOD Z, et al. CryptoGA: a cryptosystem based on genetic algorithm for cloud data security [J]. Cluster Computing, 2021, 24(2):739-752.

LIU Xuanyu, born in 1998, postgraduate. Her main research interests include cloud computing security and cyberspace security.



ZHANG Shuai, born in 1994, Ph.D, research assistant. His main research interests include cloud computing security and cyberspace security.

（责任编辑：杨雪敏）