拟态多执行体调度算法研究进展

朱正彬，刘勤让，刘冬培，王崇
（信息工程大学信息技术研究所，河南 郑州 450002）

摘 要：拟态防御是一种基于动态异构冗余架构的新型主动防御技术，具有内在不确定、异构、冗余及负反馈等特性，从而能显著提高系统稳健性和安全性。其中多执行体调度算法是拟态防御技术的关键，其优劣直接影响拟态系统抵抗基于已知或未知漏洞后门攻击的能力。基于此，首先介绍了拟态调度算法技术和目标，然后从调度对象、调度数量及调度时机这 3 个方面对调度算法研究现状进行了分析总结，最后展望了拟态调度算法未来的研究方向与趋势。

关键词：拟态防御；主动防御；拟态调度；动态异构冗余
中图分类号：TP309
文献标识码：A
DOI: 10.11959/j.issn.1000−436x.2021072

Research progress of mimic multi-execution scheduling algorithm

ZHU Zhengbin, LIU Qinrang, LIU Dongpei, WANG Chong
Institute of Information Technology, Information Engineering University, Zhengzhou 450002, China

Abstract: Mimic defense is the new active defense technology based on the dynamic heterogeneous redundant architecture. With inherent uncertainty, heterogeneous, redundant and negative feedback features, it can significantly improve the robustness and security of system. Among them, the scheduling algorithm is the key to mimic defense technology, which advantages and disadvantages directly affect the ability of system to resist attacks based on known or unknown vulnerabilities. Based on this, the principle and goal of mimic scheduling algorithm were firstly introduced. Then the state-of-the-art of mimic scheduling algorithms were analyzed and summarized from three aspects, such as scheduling object, scheduling quantity and scheduling timing. Finally, the future research direction and trend of mimic scheduling algorithms were prospected.

Keywords: mimic defense, active defense, mimic scheduling, dynamic heterogeneous redundant

1 引言

随着互联网技术的不断发展和更深层次的应用，当今社会已进入“互联网+”和“万物互联”时代，网络遍及人们生活的各个角落。与此同时，网络空间时刻存在未知漏洞和后门等不确定性威胁，使恶意攻击者利用少量的资源或代价就能侵犯个人乃至公众的隐私权,造成当今网络空间易攻难守的非对称局面。传统网络安全技术主要通过“亡羊补牢式”的策略来防范网络中频繁出现的各种网络威胁，如防火墙[1]、入侵检测系统（IDS, intrusion detection system）[2]、入侵防御系统（IPS, intrusion prevention system）[3]等，同时，漏洞挖掘[4]、特征提取[5]、蜜罐技术[6]、沙箱技术[7]等防御手段主要基于攻击手段和行为特征等先验知识精确获得。但现实生活中对未知的攻击无法精确感知，同时，对所有软硬件漏洞后门无法穷尽。据研究统计，平均 1 000～1 500 行代码中程序员就会无意留下一个漏洞[8]，系统漏洞无法避免且容易被攻击者发现并加以利用，同时，人为预留后门也存在极

收稿日期：2020−12−17；**修回日期：**2021−03−17
基金项目：国家核高基重大专项基金资助项目（No.2017ZX01030301）
Foundation Item: The National Nuclear Foundation Major Special Fund (No.2017ZX01030301)

大的安全威胁。

针对网络空间易攻难守的非对称局面，许多机构和学者提出了网络防御新思想，不再一味地追求精确已知的漏洞后门，转而采取动态、容错的新型网络空间主动防御技术，如可信计算[9]、定制可信空间[10]、移动目标防御[11-12]等。可信计算从芯片、硬件结构和操作系统等硬件底层做起，提供系统的可靠性、可用性、信息和行为安全性。定制可信空间致力于制定可信规则、可测量指标来创建灵活、分布式的环境支撑网络中空间的各种行为。移动目标防御（MTD, moving target defense）采用有效地址突变[13]、IP 地址随机化[14]、端口随机化[15]、加密随机化[16]等多样性技术增强系统脆弱面的不确定性、动态性，有效降低系统脆弱性暴露及被攻击的机会，增加攻击者扫描攻击难度，目前，该技术已成功应用于软件定义网络（SDN, software define network）[17-18]、云计算[19]等。但目前 MTD 技术发展面临许多问题，例如缺乏一定的效能评估机制，系统开销过高影响服务性能以及虚拟环境中移动目标的安全和弹性技术等问题。

基于网络主动防御思想及 MTD 技术，参考自然界中拟态章鱼能根据不同环境调节自身色彩、行为等来隐藏自己原本特征以此来躲避风险。我国邬江兴院士进一步提出拟态防御理论[20]，通过向系统引入动态、异构、冗余等特性增强系统广义稳健性和内生安全性。相比 MTD 技术，拟态防御技术通过引入仲裁和负反馈机制，使系统在具有内在攻击面[21]不确定性的同时，根据仲裁信息有针对性地调整系统内部结构，对外呈现动态性和广义不确定性，极大增加了攻击者的攻击难度，目前，拟态防御技术已成功应用于路由器[22]、交换机[23]、SDN[24]等。同时，国家科技部联合测试证明[20]，在功能等价异构冗余的多维动态重构机制作用下，网络空间拟态防御（CMD, cyberspace mimic defense）几乎不可能实现可靠、持续的协同逃逸，实验结果表明对于未知的漏洞后门有很好的防御效果。拟态防御主要实现机制包括构造效应与功能融合、策略调度、拟态裁决、负反馈控制及执行体清洗恢复等，其中拟态调度是实现拟态防御的关键一环，其基本功能是根据历史表现和负反馈信息选择或更换服务集中的执行体，实现执行体的替换、下线、转移服务等操作使拟态括号内部特征不可预测更多样化。本文在介绍拟态相关研究的同时，重点关注拟态调度相关算法的研究。

2 拟态调度架构

拟态防御基于动态异构冗余架构（DHR, dynamic heterogeneous redundant）[25]构建动态、异构、冗余且具有负反馈特性的系统和运行机制，结合仲裁和多执行体调度策略实现对系统漏洞和后门的容错。如图 1[20]所示，拟态防御主要对具有信息“输入–处理–输出”（IPO, input processing output）系统模型的网络攻击有很好的抑制作用，例如 SDN

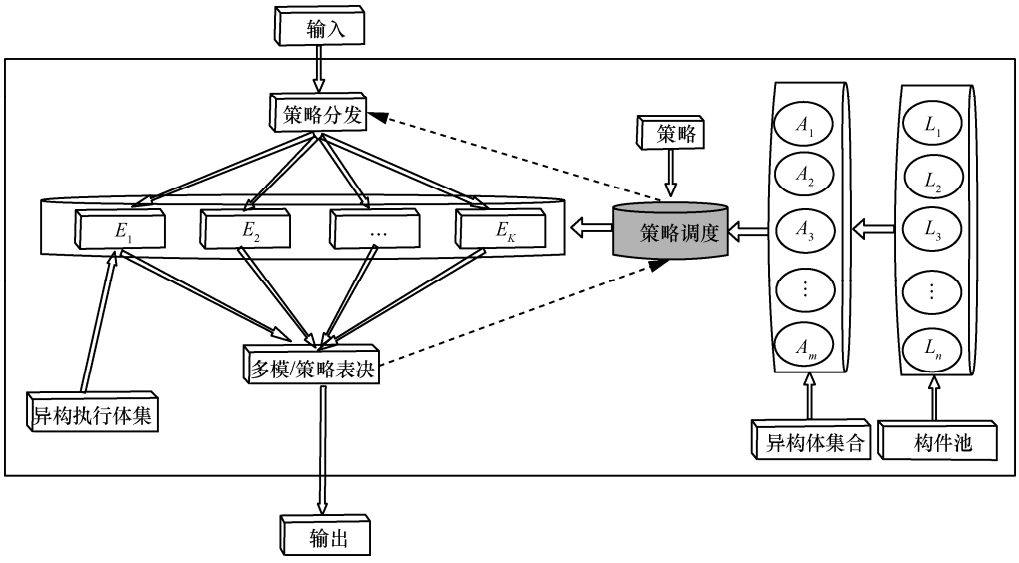

图 1 拟态防御架构

中篡改流表、控制层拒绝服务攻击、分布式数据存储数据篡改攻击等。拟态防御架构[26]主要包括输入代理即策略分发、异构执行体池、异构构件池、拟态调度器即策略调度、在线异构执行体集以及拟态多模裁决器等。其中异构执行体是结构相异功能相同的等价执行体，当所有在线执行体正常工作或未被攻击成功时接收相同的命令其输出相同，当某一执行体被攻击成功时导致其输出与其他执行体不一致，理论实践证明大多数执行体输出相同错误结果是小概率事件。输入代理负责将系统输入数据复制策略分发，即将输入数据复制成 k 份分发给 k 个功能等价结构相异的执行体集，各在线异构执行体并行执行，并将结果发送给拟态裁决器；拟态裁决器采用全体一致表决算法[27]、多数表决算法[28]、最大似然投票[29]、一致性表决算法[30]等，计算各异构执行体输出来产生最终输出结果。同时拟态裁决器将各执行体的状态反馈给调度器进行策略调度，调度器决定是否需要根据当前态势使用特定的调度算法从异构体集合中选择上线执行体，并对下线执行体进行清洗恢复等操作。例如针对 SDN 中流表篡改防御，拟态 SDN 防御系统冗余控制器假设三模冗余，即 3 个结构相异功能相同的控制器（具有相同的漏洞且被攻击输出相同的错误结果属于小概率事件），当控制器接收请求需要向交换机下发流表时，某一控制器由于存在漏洞被攻击者攻击成功然后下发错误的流表，但其他 2 个控制器正常工作输出的是正常流表。当 3 个控制器的结果到达裁决器时，根据多数表决算法即选择各执行体输出结果中占多数的结果输出，上述有 2 个正常相同结果和一个错误结果，根据多数表决即输出正确结果，然后裁决反馈错误控制器并下线清洗重新调度控制器上线。同时由于存在相同漏洞的 2 个或 3 个执行体被攻破而造成瞬时逃逸的概率极低，属于小概率事件。综上所述，拟态防御很好地抑制了基于 SDN 控制器漏洞的流表篡改攻击。拟态架构的异构性、动态性、冗余性以及负反馈特性使系统在时间和空间上具备不确定性，攻击者难以掌握系统的脆弱点，进而具备内生防御特性和天然免疫能力，拟态防御技术有望从根本上摆脱目前网络空间“易攻难守”的困局。

拟态防御技术注重拟态架构与原有系统的有机结合，使拟态系统天然具有防御基于已知或未知漏洞后门攻击的内生安全特性。多执行体调度算法是实现拟态防御的关键一环，执行体调度使系统保持高动态性和不确定性，能够避免攻击者长时间探测和协同攻击，造成瞬时逃逸的发生。调度策略流程如图 2 所示，具体分为以下 3 个步骤。

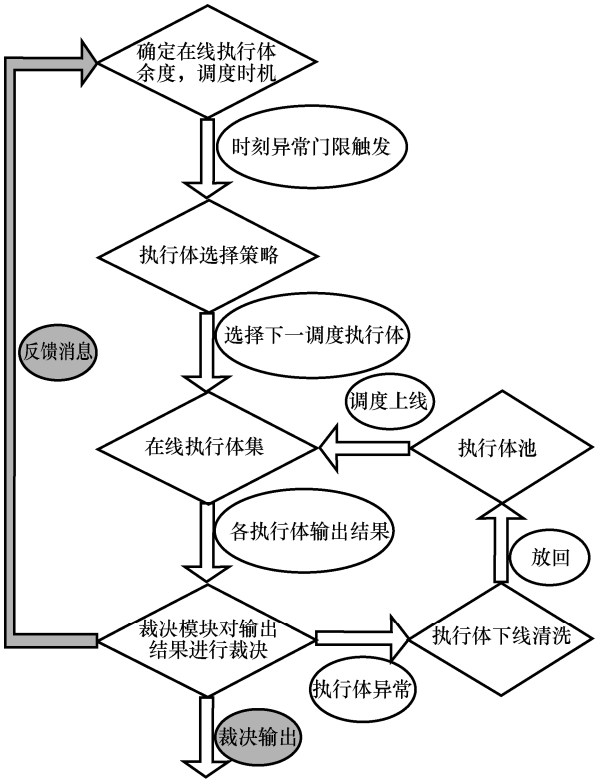

图 2　调度策略流程

1) 调度策略根据反馈信息确定在线执行体余度、调度时机，依据选择策略选取执行体上线。

2) 确定在线执行体变换门限，不定时替换异常在线执行体，下线执行体清洗。

3) 输出裁决结果，根据裁决信息反馈确定下一次变换时机和执行体冗余度。

现有主副版本容错调度[31-32]算法主要用于处理确定任务的调度进而实现容错，N-版本容错（NVT, N-version tolerance）[33]主要基于同构冗余执行体来容忍随机扰动或偶然失效，拜占庭容错（BFT, Byzantine fault tolerance）[34-35]基于冗余架构容忍拜占庭式错误，在部分执行体错误的情况下仍可以输出正确结果。上述 3 种容错方式对于系统的随机失效或非特定攻击实现了很好的容错效果，但对于特定的漏洞后门攻击防御效果不明显。拟态多执行体调度算法[36]主要是从动态异构冗余架构出发，针对基于漏洞后门或病毒木马攻击所提出的一系列调度算法。根据调度算法的目标和实现过程，可以从

以下 5 个方面对调度算法进行评估。

1) 动态性

拟态调度算法使系统内在不确定变换，从而具有动态性，即对外呈现测不准效应。文献[37]提出算法动态性可以体现在调度方案的平均周期，即相同调度方案之间产生的时间间隔。理想的调度方案要求执行体集足够大，完全相同的调度方案几乎不可能产生，现实中由于执行体余度的限制，调度方案必然存在重复的可能性。在不影响系统原本功能的同时，尽可能追求大的调度周期即提升系统动态性，是衡量某一调度算法的主要指标。

2) 可信任度

拟态容错主要研究拟态架构能否在偶然或恶意失效情况下连续、可靠、正常的执行。在拟态系统中各执行体的可信任度是系统可靠性的基础，不同的调度策略是系统可靠性的重要组成部分。对各执行体安全性量化以及系统整体性可信任度研究是调度算法亟待解决的问题。

3) 异构度

拟态架构要求功能等价、结构相异的在线执行体集，执行体间相异度越大，存在共有漏洞的可能性就越低，被攻击者利用同一漏洞攻破造成瞬时逃逸的概率就越小，执行体集相异性能显著提高系统安全性[38]。调度上线功能等价执行体异构度越大，系统安全增益越大，拟态容错调度算法追求最大执行体异构度，以免系统发生瞬时逃逸。

4) 系统开销

理想的调度算法单纯基于系统安全性考虑，缺乏对系统开销及时间代价的考虑。现实研究中如果仅单纯追求系统安全性，往往会花费极大的代价，而只能得到很少的系统安全增益。系统开销是衡量调度算法的重要指标，在追求系统安全性的同时，可以减小运算复杂度、降低时间空间代价，进而降低系统整体开销。

5) 服务质量

拟态架构期望在不影响乃至提高系统原有服务功能的基础上，使系统具备内生安全特性。但往往拟态防御机制的引入会对系统本身的服务属性造成一定的负面影响。考虑调度算法的同时要综合考虑系统的服务质量，在追求系统整体安全性最大化的同时，不降低甚至能提升系统本来的服务质量。

3 拟态调度算法

拟态调度算法主要从动态、异构、冗余出发并结合负反馈特性，实现拟态系统的高稳健性和安全性。目前，还没有相关调度算法全面地总结评估工作，经过阅读研究相关文献，本节对现有拟态调度算法进行了整理归纳，包括各调度算法的调度指标、执行效率等进行了分析比对，并对其优缺点进行了归纳。本节综合分析各调度算法的主要调度思想，从调度对象、调度时机及调度数量 3 个方面出发详细论述现有调度算法。

3.1 调度对象

拟态架构要求功能等价、结构相异的在线异构执行体集，在线执行体集相异度越大，存在共有漏洞的可能性就越低，被同一漏洞攻破造成瞬时逃逸的概率就越小。因此现有大多数调度算法集中于度量软件异构度并结合其他评价指标，从时间和空间 2 个维度出发，期望得到最大的在线执行体集异构度，避免瞬时逃逸的发生。软件异构性是整体考虑执行体异构性；组件异构度是更细粒度地将执行体划分为不同的组件，再进行组件间异构度计算。其中操作系统异构性是从系统组件出发，在计算执行体异构性时操作系统是组成该执行体的一种组件，结合执行体其他组件根据所提算法进行计算；算法层面异构性体现在计算异构度原理的不同，主要考虑从执行体和执行体集异构度具体计算方法出发，包括软件异构度、组件异构度以及基于 MOSS 度量等计算方法，已被具体考虑进下述 3 组度量方法。但不同算法在不同应用场景下所考虑的执行体组件构成有所不同。具体调度算法特点及应用如表 1 所示。

3.1.1 基于软件异构度度量

基于对软件异构度的量化，文献[39]提出最长相异性距离组件选择（MD, maximum dissimilarity）算法和最佳平均相异距离的软件组件选择（OMD, optimal mean dissimilarity）算法。MD 算法在选择软件时始终选取相异距离最长的组件，OMD 算法选取具有最佳平均相异距离的软件组件，但该算法仅考虑内部组件之间的相异距离总和，可能会与平均相异距离存在矛盾，也可能找不到足够的组件组成相异性系统，对系统阈值设置要求比较高且缺乏动态性。

由于 MD 算法和 OMD 算法缺乏对于历史信息

表 1　　基于调度对象算法优缺点及应用场景

出发点	算法	优缺点	应用场景
基于软件异构度度量	基于历史信息负反馈算法	考虑历史信息和负反馈，缺乏对异构度的考量	SDN
	基于正态分布算法	动态性、可控性好，但复杂度高	—
	动态自学习调度算法	考虑了历史信息和执行体当前安全性并自适应迭代，但计算复杂度高	SDN
	基于信誉度和相异度自适应调度算法	量化了信誉度及更新准则，缺乏细粒度的异构度量化公式	SDN
基于异构体组件度量	随机种子最小相似度算法	动态性好，执行体异构度量化更细粒度，但缺乏对负反馈特性	—
	基于 BSG 博弈算法	结合 BSG 博弈模型数学量化执行体间异构度，但计算复杂度高	Web 服务器
	随机种子调度算法	细粒度量化执行体集异构性，结合具体系统服务质量，需进一步确定权重	Web 服务器
	基于优先级和时间片调度算法	动态性高，从时间和空间 2 个维度考虑，缺乏对历史信息的考虑	—
基于 MOSS 度量	负载感知调度算法	考虑了执行体集安全性和负载，可扩展性好，但复杂度高	SDN

的考虑，吕迎迎等[40]提出基于历史信息的负反馈调度算法，利用系统监测得到的探测数据进行非一致性检验和非重复检验，分析该攻击类型进而根据不同攻击类型选择最合理的调度。仿真实验证明基于历史信息的负反馈调度算法较传统静态随机调度算法失效率在非均匀攻击时可降低 50%，在非重复攻击时可降低 58%。但该算法仅考虑外部探测对执行体的影响，没有进一步研究上线执行体间的异构度和具有同一漏洞的概率。

为了使执行体调度对外呈现动态性的同时具备内部可控性，张震骁[41]提出基于正态分布的拟态防御动态调度策略。采用序号系数 n、安全系数 s、时间系数 t、人工控制系数 a、综合系数 c 和概率系数 p 这 6 个异构执行体属性，计算某一执行体 n 被选中的概率为

$$p_n = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{c_n^2}{2}} \tag{1}$$

基于正态分布的动态调度算法以正态分布函数为载体来计算某一执行体被调度具体概率，以达到对外呈现动态测不准，对内概率可控的广义调度算法，具有很好的随机性和概率可控性。但该算法时间复杂度较高，其载体函数值得进一步优化。

由于缺乏对执行体当前安全性的考虑，Li 等[42]从系统安全性角度出发提出了一种 SDN 多控制器的动态自学习调度算法。通过控制器历史表现，评估当前时刻该控制器的可靠性，控制器组 S_i 可靠性可表示为

$$Q(Z)=\begin{cases} -\eta(Z)^2 \dfrac{F(Z)}{U(Z)}, U(Z)\neq 0 \\ 0, U(Z)=0 \end{cases} \tag{2}$$

其中，$F(Z)/U(Z)$ 表示工作中平均故障率，$\eta(Z)$ 表示集合 S 中元素的个数。根据历史表现自适应更新迭代，同时考虑负载约束利用贪心式启发算法（GA, greedy algorithm）求解出最优化调度集。该算法结合历史信息具备一定的负反馈特性，实现了对执行体当前安全度的考量，但在控制器增多时，该算法复杂度骤增且缺乏进一步的实验验证。

通过引入执行体信誉度的负反馈动态更新模块，沈丛麒等[43]提出了基于信誉度和相异度的自适应控制器调度算法，定义了信誉度动态更新准则为

$$r_i(t)=\begin{cases} \left|T_j(t-1)\right| r_i(t-1)+\delta\left(\left|\gamma(t)-y_j(t)\right|\leqslant \eta_y\right),\ i\in S(t) \\ r_i(t-1),\ i\notin S(t) \end{cases} \tag{3}$$

其中，$r_i(t)$ 表示 t 时刻执行体 i 的信誉度。此外，还从代码级、模块级、传输级和运行级 4 个层次计算执行体间相异程度并赋予各自不同的权重。相较于定期轮询清洗，文献[43]实现了较低的开销及较高的安全性。

3.1.2　基于异构体组件度量

出于对执行体间相似性更细粒度的研究，刘勤让等[37]提出了一种新的冗余体间相似度的量化方法，通过衡量各组件间的相似度进而量化 r 余度冗余体集合整体相似度为

$$S\big|_{\Omega'} = \frac{1}{C_r^2} \sum_{i=1}^{r-1} \sum_{j=i+1}^{r} \sum_{i=1}^{m} \xi_l L_l^j Y_l L_l^{j\mathrm{T}} \tag{4}$$

其中，L_l^j 表示执行体 P_j 中第 l 个组件对应的特征向量，ξ_l 表示系统组件相似度权重，Y_l 表示组件特征相似度矩阵。通过设定各执行体间最小相似度阈

值，并结合最小相似度算法提出随机种子最小相似度（RSMS, random seed minimum similarity）算法。该算法在平均调度周期较 MD 算法提升了 300%，平均失效率较随机调度算法减低 67.32%。但该算法在执行体数量较少时动态性值得进一步研究，同时缺乏对历史裁决信息的考虑。

王晓梅等[44]采取对比整体差异性的方式，将线下执行体和在线执行体的差异性求解均值和方差，提出基于贝叶斯−斯塔克尔伯格博弈（BSG, Bayesian Stackelberg game）的拟态 Web 服务器调度算法。其中，量化执行体 C_i 间的差异性为

$$W_{ij} = C_{ij} P^{\mathrm{T}} \tag{5}$$

其中，C_{ij} 表示执行体 C_i 到 C_j 软件栈各层的差异性，P^{T} 表示执行体软件栈各层差异性的加权系数。通过求解均值方差再结合 BSG 博弈收益函数，量化新的异构执行体间异构度，通过 BSG 策略增强了服务器的动态性和随机性。但 BSG 算法复杂度较高，需进一步优化，同时对攻击者类型和攻击方式统计缺乏一定的理论判别。李传煌等[45]自定义异构元素比较函数

$$w(x,y) = \begin{cases} 1 & , \quad x = y \\ 0.1 & , \quad x \neq y \end{cases} \tag{6}$$

通过两两执行体各组件对比取值，再结合该组件权重得出两执行体间异构系数。该调度算法基于异构性研究，但考虑方面过于简单，没有体现负反馈特性。

文献[46]提出多系统异构性可通过复杂性、差异性来衡量，即

$$\text{HETEROGENEITY} = \text{COMPLEXITY} \cdot \text{DISPARITY}$$

基于多系统异构性研究，张杰鑫等[47]采用上述异构性量化方法，并结合二次熵对执行体集差异性进行量化，最终通过 M 类构件集的异构性来计算执行体集异构性

$$H_A = \sum_{k=1}^{M} \left(1 - \sum_{i=1}^{s} P_{ki}^2\right) \left(\sum_{i=1}^{s} \sum_{j=1}^{s} d_{kij}^2 P_{ki} P_{kj}\right) \tag{7}$$

其中，d_{kij} 表示构件集 k 中构件 i 和 j 之间的差异性，P_{ki} 和 P_{kj} 分别表示构件 i 和 j 的丰富度。结合 Web 服务质量，文献[47]提出了基于最大异构性和 Web 服务质量的随机种子调度（RSMHQ, random seed based on maximum heterogeneous and Web QoS）算法，Web 服务质量和综合评价指标较随机调度算法分别提升了 15.32%和 37.45%，有效实现了安全性和服务质量的平衡。在具体实际应用环境中，可以根据优化算法进一步确定安全性和服务质量权重以到达最佳平衡效果。

不同于上述执行体异构度量化方法，文献[48]采用共有漏洞指标和共享代码数量分析度量执行体间的相似程度。普黎明等[49]采用文献[48]在时间和空间 2 个维度来衡量面向拟态云服务的异构执行体相似度，提出基于优先级和时间片的执行体调度（PSPT, pool scheduling based on priority and time slice）算法。该算法通过定义时间权重因子 α 和空间权重因子 β，基于执行体共有漏洞指标定义执行体相似度为

$$\begin{aligned} \mathrm{CVI}_{ly}(C_{lp}, C_{lq}) &= \sum_{i-y-\text{years}+1}^{y} \alpha_i v_i (C_{lp}, C_{lq}) \\ \mathrm{CVI}_{y}(E_j, E_k) &= \sum_{l-1}^{\text{tiers}} \beta_l \mathrm{CVI}_{ly}(C_{lp}, C_{lq}) \end{aligned} \tag{8}$$

其中，CVI_{ly} 表示同一种类构件基于共有漏洞的相似程度，CVI_{y} 表示执行体间的相似度。PSPT 算法实现了很好的动态性，在线执行体集平均调度周期约是 RSMS 算法的 1 617 倍，且时间复杂度为 $O(1)$，实现了很好的系统动态性和线性算法复杂度。同样，Wu 等[50]基于执行体共有漏洞采用 Jeccard 距离描述任意两执行体 E_i, E_j 间的异构属性为

$$\mathrm{He}_{ij} = \mathrm{He}(E_i, E_j) = 1 - \frac{\left|V(E_i) \cap V(E_j)\right|}{\left|V(E_i) \cap \Gamma(E_j)\right|} \tag{9}$$

其中，$V(E_i)$ 为执行体 E_i 的漏洞集合。结合综合调度指标 CS，文献[50]提出一种面向拟态防御系统的基于执行体异构度、性能和历史置信度的随机种子调度（RSMHQH, random seed maximum heterogeneity quality history）算法，相较于随机调度算法，该算法异构度提升了 88.68%，系统性能提升了 20.80%，综合指标 CS 提升了 42.59%。

3.1.3 基于 MOSS 度量

出于与上述异构度衡量的不同，Qiu 等[51]提出软件相似度度量（MOSS, measures of software similarity）方法，该方法利用类图的结构相似性和属性相似性，将 2 种相似性结合到迭代更新过程中计算软件相似性得分。综合考虑执行体负载，顾泽宇等[52]基于 MOSS 算法衡量系统间异构程度，并采用

凹形指数函数描述攻击成功概率 $p(v(C_{ei},t))=\int\lambda e^{-\lambda t}dt$。最大执行体集的安全系数和最小调度体集负载方差分别为

$$\max \xi(C_eS,t)=\frac{1}{m}\sum_{i\in C_eS}\xi(t)_i$$

$$\min l(C_sS,t)=\frac{1}{h}\sum_{i\in C_sS}(l(t)_j-\overline{l(t)})^2 \tag{10}$$

其中，$\xi(t)_i$ 表示集合 C_eS_i 中任意元素 C_{ei} 在时间段 τ 内的安全系数，$l(t)$ 表示调度体负载。最后提出负载感知安全调度算法（LA-SSA, load aware security scheduling algorithm）确定执行体调度策略，该算法较安全优先调度算法（SPSA, security priority scheduling algorithm）实现了很好的系统安全增益，同时负载均衡性优于 SPSA，在一定范围内有效解决了系统安全性与计算性能的平衡问题。

高明等[53]提出了一种基于拟态防御的差异化反馈调度判决算法，利用 MOSS 算法得到执行体间异构度为 $\sigma\in[0,1]$，同时，定义执行体集异构度为

$$\sigma^*=\frac{1}{2m}\sum_{i}^{m}\sum_{i}^{j}\sigma(E_i,E_j) \tag{11}$$

其中，$\sigma(E_i,E_j)$ 表示两执行体 E_i,E_j 的异构度。根据历史判决器反馈结果量化执行体安全防御系数，以形式化数学推导最优化问题得出最后调度结果，更精确地推导执行体集的相似度，有 3 个执行体时系统输出异常率平均值为 0.105 8，有 5 个执行体时仅为 0.067 3，实现了很好的安全性。该算法的最优化算法可进一步研究，优化迭代确定异构度和执行体安全系数的广义平衡。

3.2　调度时机

调度时机是实现系统动态性、对外呈现测不准效应的重要因素，调度时机问题是如何选择一个最佳的在线执行体集变换时间点，大多数采用固定时间间隔、固定异常触发次数等来进行执行体集变换。基于对调度时间问题的研究，Lu 等[54]刻画了一种闭环控制器调度安全流程，将动态调度的时间问题建模为随机理论中的更新过程，提出了一种最优调度算法（OSA, optimal scheduling algorithm）确定调度时间，即根据输入的调度代价、攻击损耗以及攻击分布函数计算最佳调度时间。定义单位代价为

$$\gamma(T_i^d)=\frac{\mathrm{E}(R_i)}{\mathrm{E}(X_i)}=\frac{C_s^d+A_{\mathrm{loss}}^dF(T_i^d)}{\int_0^{T_i^d}xF(x)dx+T_i^d(1-F(T_i^d))} \tag{12}$$

其中，$\mathrm{E}(R_i)$ 和 $\mathrm{E}(X_i)$ 分别表示第 i 次调度的总代价期望值和所用时间期望值，T_i^d 表示计算防御者第 i 次调度的时间间隔。调度目标是寻求第 i 次调度最佳时间 T_i^d 以最小化单位代价 $\gamma(T_i^d)$。OSA 综合权衡执行开销和攻击损耗这 2 个重要决定因素，较固定周期与随机调度算法，实现了较小的调度代价且调度算法平均运行时间仅为 1.91 s，在一定程度上解决了调度过程中时机选择的问题，但缺乏对执行体本身安全性、异构度等方面的考虑。

通过对异常门限 S 和调度周期 T 的综合考虑，Guo 等[55]引入滑动窗口机制提出基于滑动窗口模型的调度序列控制方法。异常门限 S 和调度周期 T 是基于内部资源和外部攻击将时间和门限耦合联动控制的调度过程。采用 Δt_i 与 s_i 窗口共同驱动

$$e_1=(t-t_i\geqslant s_i\Delta t_i)\bigcap(c_t>0)$$
$$e_2=(\varphi_i\geqslant s_i)\bigcap(c_t>0) \tag{13}$$

其中，c_t 表示可用调度容量，φ_t 表示异常反馈实时次数。该算法采用异步并行方式使调度模块和窗口滑动模块、时间复杂度和空间复杂度低。从时间层面考虑调度，该算法能够在不同场景下通过调度参数调整提升系统安全性、高效性和稳健性，具有很好的自适应能力。但该算法仅考虑一次替换单一执行体上线及随机选择，有待进一步研究。上述基于调度时机算法的优缺点及应用场景如表 2 所示。

表 2　基于调度时机算法的优缺点及应用场景

算法	优缺点	应用场景
最优调度时间算法	权衡执行开销和攻击损耗，调度代价低，但复杂度高	SDN
基于滑动窗口模型算法	综合考虑了调度门限和周期，动态性较高，复杂度低，但缺乏对多执行体调度的考虑	分布式存储

3.3　调度数量

拟态容错主要是基于 n 模冗余机制，借以裁决机制实现对执行过程中部分执行体被攻破的容错方法，大多数调度算法从动态、异构出发，通过衡量异构度和动态性增加系统稳健性和安全性，缺乏对冗余度更深层次的研究。魏帅等[56]基于安全增益、成本代价以及系统安全性等方面综合得出三模冗余容错在安全性和成本代价方面获得综合的最佳效果，但冗余度和动态性的结合缺乏更加深入的研究。

从安全性和失效率出发，Qi 等[57]在研究拟态网络操作系统（MNOS, mimic network operating sys-

tem）[58]中提出基于反馈的动态感知调度算法。根据系统所要容忍的控制器失效个数来决定下一时刻在线执行体数量，同时量化出系统失效概率模型，结合动态感知调度算法求出下一时刻运行的 NOS 集合和数量以最小化失效概率。该算法体现出负反馈特性，相较于随机切换算法以较小的代价使系统失效概率提升了近 50%，但该算法复杂度较高（$O(n^2)$），有待进一步研究。

基于对冗余度和动态性的考虑，李军飞[59]提出基于效用的动态弹性调度策略，根据当前网络环境进而确定下一步在线执行体的数量。下一次执行体调度数量 u 和下一次调度间隔 v 由当前网络中异常执行体数量 n 和故障时间 t 采用概率密度函数[60]和轮盘法[61]来确定，即

$$P(u=a)=\int_{\frac{a-1}{N}}^{\frac{a}{N}} f(n,x)\mathrm{d}x, a\in[1,N] \tag{14}$$

其中，$f(n,x)$ 是与 n 相关的概率密度函数。最后确定各个 u 值的轮盘区间为

$$\begin{cases} S(u=a)=(0,p(1)], a=1 \\ S(u=a)=\left(\sum_{N-1}^{a-1} p(u), \sum_{N-1}^{a} p(u)\right], a\in[2,N] \end{cases} \tag{15}$$

利用随机数生成 $r\in[0,1]$，确定下一次调度执行体数量。

基于判决反馈结果综合考虑动态性及冗余性，高明等[53]提出基于判决反馈的调度数量算法权衡系统代价与安全性。以执行体输出的可靠度占比为判决依据，根据 U_1 前后时刻变化更新调度个数

$$m(t)=[(1+\alpha(U_1(t-2)-U_1(t-1)))m(t-1)] \tag{16}$$

其中，U_1 表示执行体输出各类结果最大占比，$m(t-1)$ 表示上一时刻的调度个数。综合考虑系统负载，可在增加系统动态性、可靠度同时显著降低系统失效率和系统代价。但上述 3 种算法仅考虑了改变执行体数量的方法，缺乏对具体变换时间和变化条件的研究以及执行体数量变换后续裁决算法的更新，值得进一步研究。基于执行体数量调度算法的优缺点及应用场景如表 3 所示。

4 算法评估

动态性、异构度及冗余度能显著提高拟态系统安全性，现有调度算法主要从执行体调度时机、执行体选取策略以及执行体调度数量等方面实现系统动态变化，对外呈现不确定性，对内实现概率可控。在考虑系统安全性的同时，系统效率、服务质量、执行体负载等调度目标也被相关算法考虑，以期在不牺牲系统原有功能的同时，通过拟态架构使系统获得显著的安全增益。总体来说，现有调度算法一方面实现了系统安全性的显著提升，另一方面基于调度目标和系统效率的考量实现了很好的可扩展性。基于调度算法普遍考虑的调度目标，本节从动态性、可信任度（平均失效率）、异构度、系统开销以及服务质量这 5 个方面来综合比较现有算法。如表 4 所示，现有调度算法在构建评估数据集时由于执行体组件间如操作系统、处理器、应用软件、协议栈等异构度仍缺乏权威的计算标准和难度，因此大都根据经验确定在线执行体余度，然后通过 β 分布随机、MOSS 工具生成执行体间异构度，但不同算法所做实验次数不同，进而异构度衡量数值存在差异。为了使数据直观，本文在实验原理数据的基础上进行异构度值归一化；系统开销通过数学理论分析得到时间（空间）复杂度；动态性即调度周期实验通过理论分析或蒙特卡洛法观察调度周期，本节定义动态性衡量标准为平均调度周期大于 80 为高，50～80 为中高，20～50 为中，5～20 为中低，小于 5 为低。文献[34]通过假设异构执行体失效情况服从 0-1 分布，从而以一定的概率计算出随机调度算法、MD 算法、OMD 算法及 RSMS 算法的平均失效率。综上，各调度算法实验数据集都基于上述方式得出如下结果。

1) 系统开销：表 4 中调度算法使用数学理论分析时间（空间）复杂度。

2) 动态性：文献[41,44]通过分析调度算法决策结果的随机性理论分析其动态性；文献[39,47,49-50]等采用蒙特卡洛法进行重复实验 100 次，根据相同

表 3 基于执行体数量调度算法的优缺点及应用场景

算法	优缺点	应用场景
基于反馈的动态感知调度算法	动态性好，失效概率低，但复杂度高	网络操作系统
基于效用的动态弹性调度算法	动态性好，但缺乏对上线执行体异构度考虑，复杂度高	SDN
基于判决反馈调度算法	动态性好，系统失效率和代价低，但缺乏对变换时间和变化条件的研究	SDN

表 4　　算法综合性对比

调度算法	动态性	平均失效率	异构度	系统开销	服务质量
MD	★	$1.141\,9\times10^{-4}$	0.114 3	$O(1)$	—
OMD	★	$3.598\,9\times10^{-4}$	0.218 3	$O(1)$	0.450 3
随机调度	★★★★★	$7.664\,7\times10^{-4}$	0.272 3	$O(1)$	—
RSMS	★★	$2.766\,4\times10^{-4}$	0.155	$O(n)$	—
PSPT	★★★★★	—	0.249	$O(1)$	—
RSMHQ	★★	—	0.376 8	$O(n)$	0.519 3
RSMHQH	★★	—	—	$O(1)$	—
基于正态分布	★★★★	—	—	$O(2^n)$	—
基于 BSG	★★	—	MOSS	$O(n)$	—
基于反馈判决	★★★	—	MOSS	$O(n)$	—
基于自学习	★★★	—	—	$O(2^n)$	—
基于安全策略	★★	—	—	$O(n)$	—
滑动窗口	★★★★	—	—	$O(n)$	—

注：★表示低，★★表示中低，★★★表示中，★★★★表示中高，★★★★★表示高；MOSS 表示用 MOSS 方法衡量执行体间的异构度，—表示未考虑该指标。

调度方案之间的时间间隔得出其动态性。

3) 异构度：文献[37,47,49]采用自定义参数 β 分布随机生成执行体异构度数据集；文献[52-53]使用 MOSS 工具生成执行体异构度数据集。

从表 4 可以看出，前面所总结的调度目标在现有拟态调度算法中都有被考虑，大都集中于研究系统动态性、执行开销和执行体异构度量化等，但各算法在不同应用场景中考虑方面有所不同，期望在实现系统高动态性和高安全增益的同时降低系统开销和不损害系统原本服务质量。其中，PSPT 算法、随机调度算法和基于滑动窗口模型的调度算法实现了较高的动态性，RSMS 算法、RSMHQ 算法很好地量化了执行体间的异构度，RSMHQH 算法、PSPT 算法在考虑其他调度目标的同时实现了很低的算法开销。

通过形式化的数学推导和算法优化，现有调度算法很好地体现出拟态架构动态、冗余、异构以及负反馈内生安全特性，拟态系统实现了高质量的安全增益。但目前大多数算法缺乏对整体调度指标的考量以及各调度指标的综合评估，即本文所总结的调度指标的优先级以及权重。随着执行体间异构度量化遇到瓶颈，执行体调度时机和调度数量研究逐渐受到研究者重视，如何实现调度时机和执行体数量变化的有机结合以及后续裁决算法有很大的研究空间。算法复杂度和执行体异构度在特定场景中有待进一步研究优化，结合具体服务在实现系统高质量安全增益的同时不损害甚至提高实际应用的具体效能。

5　未来工作

本文综述了目前拟态调度算法的最新进展和研究成果，现有研究主要集中于考虑执行体异构度、负载率、调度时机以及调度数量等方面，在一定程度上提高了拟态系统的容错率和安全性，但目前仍存在一些问题和不足，值得进一步研究讨论。

1) 现有调度算法衡量执行体异构度大都基于执行体间共有漏洞的数量，但计算时只能依据已发现的漏洞，缺乏对执行体间未知的漏洞的考虑，同时在不同环境中各漏洞表现不一。衡量执行体异构度可以从自身结构、功能等方面结合基于度量[62]、字符串[63]、树[64]、类图[51]等方法进一步研究。

2) 现有调度算法只单方面研究调度时机或者调度数量的变化，根据不同场景变化综合考虑系统动态、异构、冗余等方面并结合最优化算法，寻求调度时机和调度数量的有机结合以实现最大的系统安全增益，有望成为下一个研究热点。

3) 现有调度算法在量化执行体集异构度时通常采用执行体间两两异构度累加的方式来计算，忽略了高阶相似性漏洞[8]的存在，如何量化执行体集高阶异构度以及与两阶异构度的分配权重有待进一步研究。

4) 结合博弈论收益模型量化调度算法的安全

收益和系统代价，当前调度算法主要是依据实验结果来衡量系统的安全增益，缺少一定的理论依据和可扩展性。在具体环境中结合博弈论建模攻击者和防御者的攻防行为求解最优调度，并通过实验结果验证更具有一般性和说服力。

6 结束语

随着网络入侵窃密事件频发，拟态防御凭借其主动防御思想在网络安全方面逐渐显示出强大的防御能力，拟态调度算法也发挥了越来越重要的作用。随着拟态技术的不断发展应用，在不同应用环境中新的拟态调度问题也不断涌现，值得进一步创新和改进。本文在现有文献的基础上，首先介绍了拟态防御架构、实现原理及其实际应用；然后以各调度算法研究点为主线并考虑算法时间先后顺序，总结提出了评估调度算法的五大指标，研究了拟态调度的工作机理及其安全机制，分析了现有各调度算法的创新与不足；最后对拟态调度的趋势和研究方向进行了展望，期望通过总结拟态调度算法的现有成果为相关研究人员提供理论参考和帮助。

参考文献：

[1] OPPLIGER R. Internet security[J]. Communications of the ACM, 1997, 40(5): 92-102.

[2] ROBERTO D P, LUIGI V M. Intrusion detection systems[M]. Berlin: Springer Science & Business Media, 2008.

[3] PANDA B K, PRADHAN M, PRADHAN S K. Intrusion prevention system[M]. Network Security Attacks and Countermeasures. IGI Global, 2016.

[4] GHAFFARIAN S M, SHAHRIARI H R. Software vulnerability analysis and discovery using machine-learning and data-mining techniques[J]. ACM Computing Surveys, 2017, 50(4): 1-36.

[5] HOSSEINI S. Fingerprint vulnerability: a survey[C]//2018 4th International Conference on Web Research. Piscataway: IEEE Press, 2018: 70-77.

[6] PERDISCI R, DAGON D, LEE W, et al. Misleading worm signature generators using deliberate noise injection[C]//2006 IEEE Symposium on Security and Privacy. Piscataway: IEEE Press, 2006: 17-31.

[7] REIS C, BARTH A, PIZANO C. Browser security: lessons from google chrome[J]. Queue, 2009, 7(5): 3-8.

[8] 魏帅, 张辉华, 苏野, 等. 基于高阶异构度的大数裁决算法及性能分析[J]. 计算机工程, 2020, 51(1): 1-7.
WEI S, ZHANG H H, SU Y, et al. Majority voting algorithm and performance analysis based on high level heterogeneity[J]. Computer Engineering, 2020, 51(1): 1-7.

[9] 沈昌祥, 张大伟, 刘吉强, 等. 可信 3.0 战略: 可信计算的革命性演变[J]. 中国工程科学, 2016, 18(6): 53-57.
SHEN C X, ZHANG D W, LIU J Q, et al. The strategy of TC 3.0: a revolutionary evolution in trusted computing[J]. Engineering Science, 2016, 18(6): 53-57.

[10] CONG J, SARKAR V, REINMAN G, et al. Customizable domain-specific computing[J]. IEEE Design & Test of Computers, 2011, 28(2): 6-15.

[11] ZHENG J J, NAMIN A S. A survey on the moving target defense strategies: an architectural perspective[J]. Journal of Computer Science and Technology, 2019, 34(1): 207-233.

[12] 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.

[13] JAFARIAN J H, AL-SHAER E, DUAN Q. An effective address mutation approach for disrupting reconnaissance attacks[J]. IEEE Transactions on Information Forensics and Security, 2015, 10(12): 2562-2577.

[14] JAFARIAN J H, AL-SHAER E, DUAN Q. Adversary-aware IP address randomization for proactive agility against sophisticated attackers[C]//2015 IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2015: 738-746.

[15] LUO Y B, WANG B S, CAI G L. Effectiveness of port hopping as a moving target defense[C]//2014 7th International Conference on Security Technology. Piscataway: IEEE Press, 2014: 7-10.

[16] AZAB M, ELTOWEISSY M. ChameleonSoft: software behavior encryption for moving target defense[J]. Mobile Networks and Applications, 2013, 18(2): 271-292.

[17] SAKIC E, ĐERIĆ N, KELLERER W. MORPH: an adaptive framework for efficient and Byzantine fault-tolerant SDN control plane[J]. IEEE Journal on Selected Areas in Communications, 2018, 36(10): 2158-2174.

[18] KAMPANAKIS P, PERROS H, BEYENE T. SDN-based solutions for moving target defense network protection[C]//Proceedings of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks. Piscataway: IEEE Press, 2014: 1-6.

[19] TORQUATO M, VIEIRA M. Moving target defense in cloud computing: a systematic mapping study[J]. Computers & Security, 2020, 92: 101742.

[20] 邬江兴. 网络空间拟态防御研究[J]. 信息安全学报, 2016, 1(4): 1-10.
WU J X. Research on cyber mimic defense[J]. Journal of Cyber Security, 2016, 1(4): 1-10.

[21] MANADHATA P K, WING J M. An attack surface metric[J]. IEEE Transactions on Software Engineering, 2011, 37(3): 371-386.

[22] 马海龙, 江逸茗, 白冰, 等. 路由器拟态防御能力测试与分析[J].信息安全学报, 2017, 2(1): 43-53.
MA H L, JIANG Y M, BAI B, et al. Tests and analyses for mimic defense ability of routers[J]. Journal of Cyber Security, 2017, 2(1): 43-53.

[23] 宋克, 刘勤让, 魏帅, 等. 基于拟态防御的以太网交换机内生安全体系结构[J]. 通信学报, 2020, 41(5): 18-26.
SONG K, LIU Q R, WEI S, et al. Endogenous security architecture of Ethernet switch based on mimic defense[J]. Journal on Communications, 2020, 41(5): 18-26.

[24] 卢振平, 陈福才, 程国振. 基于贝叶斯-斯坦科尔伯格博弈的 SDN 安全控制平面模型[J]. 网络与信息安全学报, 2017, 3(11): 40-49.
LU Z P, CHEN F C, CHENG G Z. Secure control plane for SDN using Bayesian Stackelberg games[J]. Chinese Journal of Network and Information Security, 2017, 3(11): 40-49.

[25] WANG W, LI G S, GAI K K, et al. Modelization and analysis of dynamic heterogeneous redundant system[J]. Concurrency and Computation Practice and Experience, 2020, 35(2): 35-43.

[26] HU H C, WU J X, WANG Z P, et al. Mimic defense: a designed-in cybersecurity defense framework[J]. IET Information Security, 2018, 12(3): 226-237.

[27] PARHAMI B. Voting algorithms[J]. IEEE Transactions on Reliability, 1994, 43(4): 617-629.

[28] JAMALI N, SAMMUT C. Majority voting: material classification by tactile sensing using surface texture[J]. IEEE Transactions on Robotics, 2011, 27(3): 508-521.

[29] LEUNG Y W. Maximum likelihood voting for fault-tolerant software with finite output-space[J]. IEEE Transactions on Reliability, 1995, 44(3): 419-427.

[30] MCALLISTER D F, SUN C E, VOUK M A. Reliability of voting in fault-tolerant software systems for small output-spaces[J]. IEEE Transactions on Reliability, 1990, 39(5): 524-534.

[31] REIS G A, CHANG J, VACHHARAJANI N, et al. SWIFT: software implemented fault tolerance[C]//International Symposium on Code Generation and Optimization. Piscataway: IEEE Press, 2005: 243-254.

[32] 彭浩, 陆阳, 孙峰, 等. 副版本不可抢占的全局容错调度算法[J]. 软件学报, 2016, 27(12): 3158-3171.
PENG H, LU Y, SUN F, et al. Fault tolerant global scheduling with non-preemptive backups[J]. Journal of Software, 2016, 27(12): 3158-3171.

[33] AVIZIENIS A. The N-version approach to fault-tolerant software[J]. IEEE Transactions on Software Engineering, 1985, SE-11(12): 1491-1501.

[34] CASTRO M, LISKOV B. Practical Byzantine fault tolerance and proactive recovery[J]. ACM Transactions on Computer Systems, 2002, 20(4): 398-461.

[35] VERONESE G S, CORREIA M, BESSANI A N, et al. Efficient Byzantine fault-tolerance[J]. IEEE Transactions on Computers, 2013, 62(1): 16-30.

[36] 邬江兴. 网络空间拟态防御导论[M]. 北京: 科学出版社, 2017.
WU J X. Introduction to cyberspace mimic defense[M]. Beijing: Science Press, 2017.

[37] 刘勤让, 林森杰, 顾泽宇. 面向拟态安全防御的异构功能等价体调度算法[J]. 通信学报, 2018, 39(7): 188-198.
LIU Q R, LIN S J, GU Z Y. Heterogeneous redundancies scheduling algorithm for mimic security defense[J]. Journal on Communications, 2018, 39(7): 188-198.

[38] 韩进, 臧斌宇. 软件相异性对于系统安全的有效性分析[J]. 计算机应用与软件, 2010, 27(9): 273-275, 300.
HAN J, ZANG B Y. Analyzing the effectiveness of software diversity for system security[J]. Computer Applications and Software, 2010, 27(9): 273-275, 300.

[39] 姚文斌, 杨孝宗. 相异性软件组件选择算法设计[J]. 哈尔滨工业大学学报, 2003, 35(3): 261-264.
YAO W B, YANG X Z. Design of selective algorithm for diverse software components[J]. Journal of Harbin Institute of Technology, 2003, 35(3): 261-264.

[40] 吕迎迎, 郭云飞, 王禛鹏, 等. SDN 中基于历史信息的负反馈调度算法[J]. 网络与信息安全学报, 2018, 4(6): 45-51.
LYU Y Y, GUO Y F, WANG Z P, et al. Negative feedback scheduling algorithm based on historical information in SDN[J]. Chinese Journal of Network and Information Security, 2018, 4(6): 45-51.

[41] 张震骁. 拟态防御动态调度策略研究[D]. 郑州: 郑州大学, 2018.
ZHANG Z X. Research on dynamic scheduling strategy for mi-mic defense[D]. Zhengzhou: Zhengzhou University, 2018.

[42] LI J F, WU J X, HU Y X, et al. DSL: dynamic and self-learning schedule method of multiple controllers in SDN[J]. ETRI Journal, 2017, 39(3): 364-372.

[43] 沈丛麒, 陈双喜, 吴春明, 等. 基于信誉度与相异度的自适应拟态控制器研究[J]. 通信学报, 2018, 39(S2): 173-180.
SHEN C Q, CHEN S X, WU C M, et al. Adaptive mimic defensive controller framework based on reputation and dissimilarity[J]. Journal on Communications, 2018, 39(S2): 173-180.

[44] 王晓梅, 杨文晗, 张维, 等. 基于 BSG 的拟态 Web 服务器调度策略研究[J]. 通信学报, 2018, 39(S2): 112-120.
WANG X M, YANG W H, ZHANG W, et al. Research on scheduling strategy of mimic Web server based on BSG[J]. Journal on Communications, 2018, 39(S2): 112-120.

[45] 李传煌, 任云方, 汤中运, 等. SDN 中服务部署的拟态防御方法[J]. 通信学报, 2018, 39(S2): 121-130.
LI C H, REN Y F, TANG Z Y, et al. Mimic defense method for service deployment in SDN[J]. Journal on Communications, 2018, 39(S2): 121-130.

[46] TWU P, MOSTOFI Y, EGERSTEDT M. A measure of heterogeneity in multi-agent systems[C]//2014 American Control Conference. Piscataway: IEEE Press, 2014: 3972-3977.

[47] 张杰鑫, 庞建民, 张铮, 等. 面向拟态构造 Web 服务器的执行体调度算法[J]. 计算机工程, 2019, 45(8): 14-21.
ZHANG J X, PANG J M, ZHANG Z, et al. Executors scheduling algorithm for Web server with mimic structure[J]. Computer Engineering, 2019, 45(8): 14-21.

[48] GARCIA M, BESSANI A, GASHI I, et al. Analysis of operating system diversity for intrusion tolerance[J]. Software: Practice and Experience, 2014, 44(6): 735-770.

[49] 普黎明, 刘树新, 丁瑞浩, 等. 面向拟态云服务的异构执行体调度算法[J]. 通信学报, 2020, 41(3): 17-24.
PU L M, LIU S X, DING R H, et al. Heterogeneous executor scheduling algorithm for mimic cloud service[J]. Journal on Communications, 2020, 41(3): 17-24.

[50] WU Z Q, WEI J. Heterogeneous executors scheduling algorithm for mimic defense systems[C]//2019 IEEE 2nd International Conference on Computer and Communication Engineering. Piscataway: IEEE Press, 2019: 279-284.

[51] QIU D H, LI H, SUN J L. Measuring software similarity based on structure and property of class diagram[C]//2013 Sixth International Conference on Advanced Computational Intelligence. Piscataway: IEEE Press, 2013: 75-80.

[52] 顾泽宇, 张兴明, 林森杰. 基于安全策略的负载感知动态调度机制[J]. 计算机应用, 2017, 37(11): 3304-3310.
GU Z Y, ZHANG X M, LIN S J. Load-aware dynamic scheduling mechanism based on security strategies[J]. Journal of Computer Applications, 2017, 37(11): 3304-3310.

[53] 高明, 罗锦, 周慧颖, 等. 一种基于拟态防御的差异化反馈调度判决算法[J]. 电信科学, 2020, 36(5): 73-82.
GAO M, LUO J, ZHOU H Y, et al. A differential feedback scheduling decision algorithm based on mimic defense[J]. Telecommunications Science, 2020, 36(5): 73-82.

[54] LU Z P, CHEN F C, CHENG G Z, et al. Towards a dynamic controller scheduling-timing problem in software-defined networking[J]. China

Communications, 2017, 14(10): 26-38.

[55] GUO W, WU Z Q, ZHANG F, et al. Scheduling sequence control method based on sliding window in cyberspace mimic defense[J]. IEEE Access, 2019, 8: 1517-1533.

[56] 魏帅, 于洪, 顾泽宇, 等. 面向工控领域的拟态安全处理机架构[J]. 信息安全学报, 2017, 2(1): 54-73.
WEI S, YU H, GU Z Y, et al. Architecture of mimic security processor for industry control system[J]. Journal of Cyber Security, 2017, 2(1): 54-73.

[57] QI C, WU J X, HU H C, et al. Dynamic-scheduling mechanism of controllers based on security policy in software-defined network[J]. Electronics Letters, 2016, 52(23): 1918-1920.

[58] HU H C, WANG Z P, CHENG G Z, et al. MNOS: a mimic network operating system for software defined networks[J]. IET Information Security, 2017, 11(6): 345-355.

[59] 李军飞. 软件定义网络中拟态防御的关键技术研究[D]. 郑州: 战略支援部队信息工程大学, 2019.
LI J F. Research on key technologies of mimic defense in software-defined network[D]. Zhengzhou: Information Engineering University, 2019.

[60] PARZEN E. On estimation of a probability density function and mode[J]. The Annals of Mathematical Statistics, 1962, 33(3): 1065-1076.

[61] LIPOWSKI A, LIPOWSKA D. Roulette-wheel selection via stochastic acceptance[J]. Physica A: Statistical Mechanics and Its Applications, 2012, 391(6): 2193-2196.

[62] TAMADA H. Java birthmarks: detecting the software theft[J]. IEICE Transactions on Information and Systems, 2005, 88(9): 2148-2158.

[63] PARK H, CHOI S, LIM H I, et al. Detecting code theft via a static instruction trace birthmark for Java methods[C]//2008 6th IEEE International Conference on Industrial Informatics. Piscataway: IEEE Press, 2008: 551-556.

[64] BAXTER I D, YAHIN A, MOURA L, et al. Clone detection using abstract syntax trees[C]//Proceedings of International Conference on Software Maintenance. Piscataway: IEEE Press, 1998: 368-377.

[作者简介]

朱正彬（1996− ），男，湖北荆门人，信息工程大学博士生，主要研究方向为网络空间安全、网络主动防御。

刘勤让（1975− ），男，河南商丘人，博士，信息工程大学研究员，主要研究方向为网络空间安全、宽带信息网络及芯片设计。

刘冬培（1985− ），男，湖南长沙人，博士，信息工程大学助理研究员，主要研究方向为 SoC 芯片测试与验证。

王崇（1995− ），男，河北邯郸人，信息工程大学博士生，主要研究方向为拟态防御、缓存侧信道防御。