

面向云应用的拟态云服务架构

普黎明, 卫红权, 李星, 江逸茗
(信息工程大学, 河南 郑州 450001)

摘要: 针对单执行体的云应用服务缺乏异构性和动态性, 难以应对未知漏洞和后门的安全威胁问题, 提出一种拟态云服务架构, 把云平台向用户提供的应用服务节点构造成基于拟态防御技术的服务包, 使应用服务具有拟态构造带来的内生安全特性和鲁棒性, 同时讨论了策略调度和裁决机制等两项关键的拟态云服务运行机制。经实验分析表明, 拟态云服务具有较好的安全性, 可以通过减小执行体的性能差异降低其响应时延。

关键词: 拟态云服务; 响应时延; 动态; 异构

中图分类号: TP309

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2021011

Mimic cloud service architecture for cloud applications

PU Liming, WEI Hongquan, LI Xing, JIANG Yiming
Information Engineering University, Zhengzhou 450001, China

Abstract: In order to solve the problem of the lack of heterogeneity and dynamics of cloud application services with a single executor, and the difficulty of dealing with the security threats of unknown vulnerabilities and backdoors, a mimic cloud service architecture was proposed. In this architecture, the application services provided by the cloud platform were constructed into a service package based on mimic defense technology, so that the application services had the endogenous security features and robustness brought by mimic structure. At the same time, two key mimic cloud services operating mechanism, policy scheduling and adjudication mechanism were discussed. The experimental results and analysis show that the mimic cloud service obtains better security and its response time delay can be reduced by reducing the performance difference of the executor.

Keywords: mimic cloud service, response delay, dynamic, heterogeneous

1 引言

随着云服务技术优势的体现, 越来越多的用

户选择将其业务上云, 基于云的应用服务得到快速发展。在云服务提供商向用户提供多元化、多层次不同服务的同时, 云服务面临的安全问题也

收稿日期: 2020-04-08; 修回日期: 2020-07-05

通信作者: 普黎明, plm@ndsc.com.cn

基金项目: 国家科技重大专项基金(2018ZX03002002); 国家自然科学基金(61521003)

Foundation Items: The National Science and Technology Major Project of China (2018ZX03002002), The National Natural Science Foundation of China (61521003)

论文引用格式: 普黎明, 卫红权, 李星, 等. 面向云应用的拟态云服务架构[J]. 网络与信息安全学报, 2021, 7(1): 101-112.
PU L M, WEI H Q, LI X, et al. Mimic cloud service architecture for cloud applications[J]. Chinese Journal of Network and Information Security, 2021, 7(1): 101-112.

日益增多,包括传统网络安全威胁和云服务场景带来的新挑战^[1-2]。例如,传统的 DDoS^[3]、入侵^[4]、病毒^[5]、注入^[6]等安全问题和针对云架构的虚拟机逃逸^[7-8]、资源滥用^[9]、侧信道攻击^[10-12]等。据国家互联网应急中心(CNCERT, National Internet Emergency Center)2019年上半年的统计数据显示,各类型网络安全事件中,云服务被植入后门链接数量占境内全部被植入后门链接数量的63.1%、被篡改网页数量占境内被篡改网页数量的62.5%,云服务已成为网络攻击的“重灾区”,其安全问题已经成为阻碍用户将应用迁移至云服务的重要因素之一。

云服务系统通过集成现有技术的软硬件基础设施进行构建,各部件已有的漏洞和后门加剧向云服务平台聚集,其每个层面和部件出现的漏洞和后门都将直接影响整个系统的正常安全运作,据CNCERT统计数据显示,自2014年以来,国家信息安全漏洞共享平台(CNVD, China National Vulnerability Database)收录安全漏洞数量年平均增长率为15.0%,其中2019年上半年“零日”漏洞(指CNVD收录该漏洞时还未公布补丁)收录数量同比增长34.0%。因此,漏洞和后门已成为云服务安全的重要威胁。例如,有名的“毒液(VENOM)”漏洞(编号CVE-2015-3456)^[13]可允许攻击者从受感染虚拟机中摆脱访客身份限制获取主机的代码执行权限,进而控制宿主机系统实现虚拟机逃逸,此漏洞可能危害数据中心网络中的任何一台机器,数以百万计的虚拟机很容易受到利用此漏洞的攻击。

然而,现行的云服务安全防护技术大多基于威胁特征和攻击行为感知等先验知识的被动式防御,如病毒防护技术需要预先知道病毒特征码建立病毒库^[5],入侵检测技术也需要知道威胁特征^[4,14],这类技术需要不停地找漏洞并给系统打补丁,难以有效抵御基于软硬件未知漏洞或后门等的不确定威胁。近几年出现的移动目标防御(MTD, moving target defense)技术通过使IP地址、服务端口、后端数据源或者路由等动态变化给系统带来不确定性,从而增加攻击者入侵难度^[15-18],但其面对未知后门问题依然难以解决。

网络空间拟态防御(CMD, cyber mimic

defense)技术克服了传统安全方法的诸多问题,面对未知漏洞后门、病毒木马等不确定威胁时具有显著效果,该技术以动态异构冗余(DHR, dynamical heterogeneous and redundant)为核心架构,其能够大幅提升攻击难度,增强信息系统的安全性,同时具备发现未知攻击的能力^[19]。目前,在网络空间各层面已有一些CMD技术的研究成果,如硬件层有拟态安全处理器^[20],网络层有拟态路由器、拟态SDN控制器^[21-23]、拟态域名服务器^[24]等,数据层有拟态存储系统^[25]等,应用层有拟态Web服务器^[26]等。

然而,针对云服务安全防护,还没有相应的拟态防御技术研究。本文以云平台应用服务节点的安全防护为切入点,提出面向应用的拟态云服务(MCS, mimic cloud service),对云服务节点构造基于CMD技术的拟态云服务包(MSP, mimic service package),并以服务包的形式提供该节点的业务服务。同时,云平台天然具有的异构冗余特性以及资源池化和虚拟化带来的动态性为构建MSP提供了切实可行的基础支撑条件。另外,已有的研究大多从可靠性和抗攻击能力等方面对CMD系统进行测试分析,而对用户比较关心的响应时延问题少有探讨,本文对拟态云服务的响应时延进行了实验分析。

2 相关概念和定义

本节先简要介绍DHR架构,然后对相关概念进行定义。

2.1 DHR架构

DHR架构是CMD的核心架构,该架构由输入代理、执行体池(EP, executor pool)、表决器、策略调度器、异构资源池等组成。策略调度器控制各部件协同工作;输入代理接收用户请求并向执行体池进行分发;执行体池进行请求处理并将结果送给表决器;表决器收齐各执行体的处理结果后进行一致性择多判决,最终产生唯一输出并由输出代理回送给用户^[19,27]。其架构如图1所示。

动态、异构和冗余是DHR架构的主要特性,动态性对外呈现出不确定性,使攻击者难以形成有效的探测和攻击链^[15-16];异构性减少了执行体共有漏洞和后门等威胁,降低了协同攻击的可能

性^[28]；冗余性增加了系统的容错能力，提高了系统的可靠性^[29-30]。策略调度使系统持续保持以上特性平衡稳定^[31-32]，表决器通过裁决策略和信息反馈使系统能够发现异常，并进行自我修正和恢复，进而有效抑制拟态逃逸风险^[19,33]。DHR 架构具有高容错能力，能独立提供确定或不确定威胁的防御能力，可以解决可靠和可信服务环境的鲁棒性、柔韧性和安全性问题^[19]。

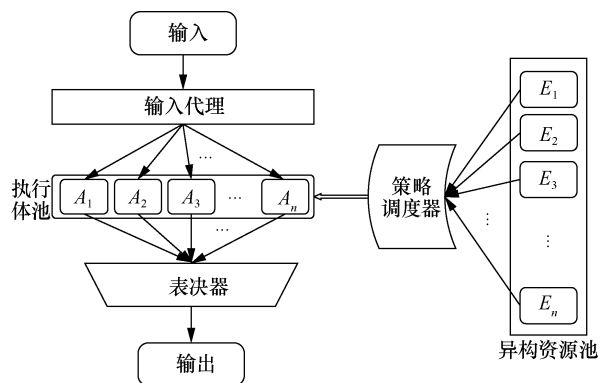


图 1 DHR 架构
Figure 1 DHR architecture

2.2 相关概念定义

定义 1 云平台

云服务的基础设施与管理平台，也是云服务（含拟态云服务）的提供方，一般由云服务提供商建设。

定义 2 管理者

云平台的用户，应用服务的开发及管理方，其使用云平台提供的拟态云服务，也可被称作应用管理者。

定义 3 用户

云应用服务的最终用户，即云应用的消费者。

定义 4 余度

执行体池中执行体的数量称作执行体冗余度，简称余度。

3 拟态云服务架构

云服务系统通常由物理资源、虚拟资源池、云管理平台、云服务平台、应用系统组成，其架构如图 2 所示。

图 2 中物理资源和虚拟资源池构成云服务基础设施层，在云管理平台的调度控制下支撑云服务平台向管理者的应用系统提供服务，云安全管

理组件提供从底层到应用层的安全管控。云服务安全通常需要提供和管理者共同负责，提供商主要对云服务基础设施及相关资源的安全负责，通常能够提供足够措施保障云平台本身的安全；管理者对数据和应用安全负责，其通常采用病毒木马防护、补丁修复、入侵检测等基于已知漏洞和后门等先验知识的防护系统进行安全保障，难以应对未知安全威胁，如“零日”漏洞和后门等。

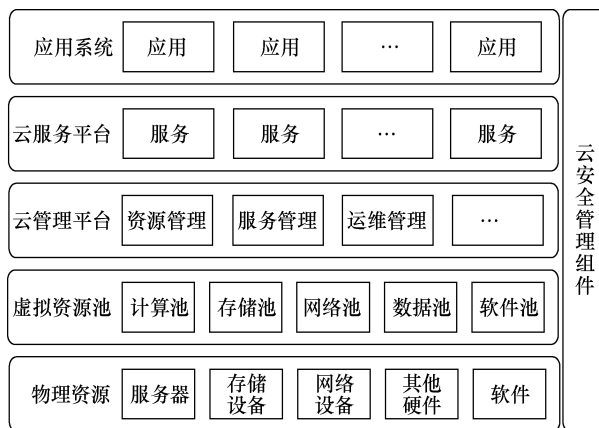


图 2 云服务架构
Figure 2 Cloud service architecture

本文提出在云服务平台，面向应用服务节点构造基于 DHR 架构的 MSP，使云平台可以向用户提供基于 CMD 技术的云服务节点，不但能够解决未知安全威胁问题，同时提高服务的鲁棒性。MSP 作为整体单元提供标准的应用服务，符合请求-响应模型的应用系统都可使用。根据文献[26,30]的研究表明，余度为 3 的执行体池较为实用，能够获得最佳性价比，本文主要针对 3 余度的情况进行相关讨论。

拟态云服务架构如图 3 所示，拟态云服务不改变原平台的系统结构，只在云平台的服务层构造满足 DHR 架构的 MSP 向应用管理者提供拟态化的云服务。云服务平台管理平面以插件的形式扩展对拟态云服务的功能支持，应用管理者在申请云服务资源时可根据需要选择拟态云服务，即选择以 MSP 的形式提供的云服务，以获得安全性和鲁棒性。

图 3 中应用系统由应用管理者开发部署，应用的后端服务由图中“服务”虚线框表示，其服务实体是拟态云服务包 MSP，一个服务由一个 MSP 来提供，服务可有多 API 接口；一个应用可以有多个 MSP，也可以只用一个 MSP。MSP 由拟态服务控制器（MSC，mimic service controller）、服务代

理 (MSA, mimic service agent)、执行体池、虚拟节点 (VN, virtual node) 及相关资源和策略组成, 构建拟态云服务的关键就是构造 MSP。

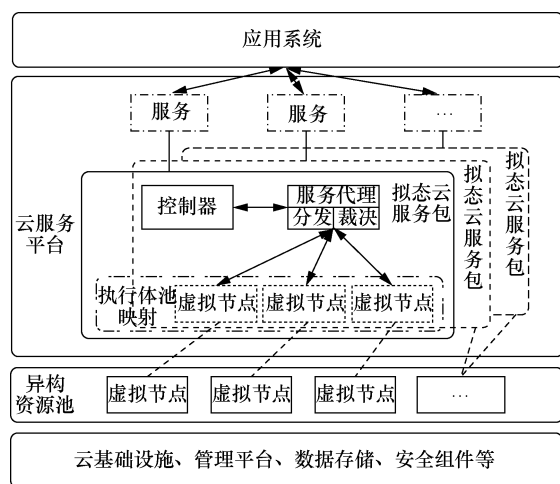


图3 拟态云服务架构示意
Figure 3 Mimic cloud service architecture

3.1 拟态服务控制器

拟态服务控制器是 MSP 的控制与调度单元, 其主要功能如下。

(1) 策略管理

负责执行体调度策略, 输出裁决、时间片、清洗等策略的管理与调用; 策略可由云平台系统预定义, 也可由应用管理者自定义或者进行参数调整; 策略是保证 MSP 正常运行的基础, 每个 MSP 都有一份策略配置表记录该 MSP 可以使用的所有策略, 由 MSC 维护。MSP 启动时、管理者下指令或者策略触发时, MSC 会把最新的策略配置表推送给 MSA。

(2) 服务代理管理

MSC 通过向 MSA 推送策略、执行体池信息等消息控制其运行, 并监测 MSA 运行状态, 发现 MSA 异常时启动恢复程序保障其持续服务。

(3) 执行体池管理

根据调度、时间片等策略以及 MSA 的反馈信息, MSC 动态调整执行体池, 并通知云平台清洗或重新配置下线的执行体, 以使异常的执行体恢复初始健康状态。

(4) 运行监控

MSC 对 MSP 的各组件以及云平台提供的关资源 (包括计算、网络 and 存储资源) 进行监测,

发现异常时向管理者和云平台报送异常消息。

MSC 由云平台提供实现, 其不参与服务业务 (请求与响应) 过程和数据的处理, 除 MSA 可以向其反馈裁决信息外, 不接收任何连接请求。控制信息由 MSC 主动推送给 MSA, 执行体的下线与清洗等需要云平台执行的操作均由 MSC 主动向其推送控制消息。通过尽可能减少 MSC 的攻击面, 提高 MSC 的安全性, 进而保障 MSP 的安全。在此, 本文假定云服务提供者已经采取足够的措施保障云基础设施的安全, 即假定云基础设施是可信系统。

3.2 拟态服务代理

拟态服务代理是请求输入和响应输出的控制单元, 其主要功能如下。

(1) 接收和分发请求

负责接收来自用户的业务请求, 并根据 MSC 提供的策略向执行体池的所有执行体分发, 此处的执行体池也是由 MSC 提供的。

(2) 裁决和输出

负责接收执行体返回的业务处理结果, 当收齐所有执行体的结果时, 根据 MSC 提供的策略进行一致性大数裁决, 当有少数不一致的结果时, 把该结果对应的执行体信息反馈给 MSC, 并把一致结果输出给用户。

MSA 由云平台提供实现, 其不参与服务业务的处理过程, 只把用户请求向执行体池中的执行体进行分发, 并对执行体返回的结果进行一致性大数裁决, 最后向用户返回唯一结果。

3.3 执行体池

执行体池是用户请求的业务处理单元, 其按照管理者的冗余度要求处理 MSA 分发的业务请求。EP 中的多个执行体同时处理业务请求, 并分别返回结果给 MSA。

EP 由 MSC 根据调度策略从异构资源池中动态生成, 其在 MSP 中体现为配置表, 该配置表描述了 EP 和异构资源池的映射关系。换句话说, EP 是一个逻辑单元, 其实际执行体节点由配置表映射到异构资源池。

图4是3冗余条件下一个执行池的映射关系示意, 图中执行体池映射表中的节点2、3、5实际是异构资源池中的节点2、3、5, MSC 根据策

略调整执行体池后,映射表中的节点将换成另外的组合,执行体池实际上就是一张逻辑配置表。

在实际运用中, MSC 根据余度要求和 MSP 异构资源池中虚拟节点的数量一次性生成一个执行体池列表,该列表包含了 MSP 可用虚拟节点在余度要求下的所有可能的执行体池组合,其列表长度为 C_n^r ,其中, n 表示 MSP 所属异构资源池中虚拟节点的数量, r 表示余度。

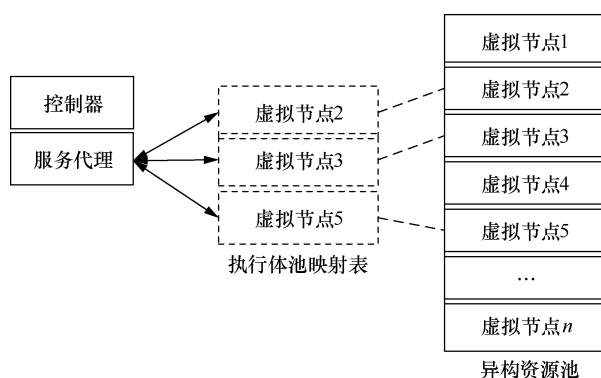


图 4 执行体池映射关系
Figure 4 Executor pool mapping

3.4 虚拟节点

虚拟节点是拟态云服务的执行体,负责对用户业务请求进行处理并返回结果。云平台根据管理者的申请提供足够的异构 VN 资源给 MSP 作为异构资源池, MSC 根据调度策略动态生成执行体池供 MSA 使用。

当虚拟节点异常(包括裁决为少数节点、宕机等)时, MSC 根据策略调整执行体池并下线该节点,而后根据策略通知云平台对该节点进行清洗(如系统还原、删除换新等)、重配置等操作,也可通知管理者进行人工干预和异常分析,解除相关问题隐患后重新上线进入资源池备用。

虚拟节点的实例可以是虚拟机或者容器,要求各节点的实例分布在不同的物理主机。另外,根据管理者需要,节点实例也可以是物理机。

虚拟节点可在多个层面提供异构性支持,如宿主机、虚拟软件、操作系统、应用软件等层面。

4 运行机制

本节分别介绍拟态云服务配置与工作流程、执行体调度机制和输出裁决机制。

4.1 拟态云服务配置

MSP 从配置生成到提供服务要经过 3 个阶段,分别如下。

1) 配置阶段:云平台根据管理者需求生成 MSP,其表现为一组配置信息,如表 1 所示。

表 1 MSP 主要配置信息
Table 1 MSP configuration information

配置项	描述
MSP_IP	云服务的 IP 地址
MSC_IP	拟态服务控制器的 IP 地址
MSA_IP	服务代理面向执行体的 IP 地址
VN_POOL	异构资源池
SCHEDULING_POLICY	预定义的调度策略
DECISION_POLICY	预定义的裁决策略

2) 启动阶段:管理者启动 MSP,云平台根据配置信息生成 MSP 实例, MSC、MSA 及所有 VN 资源启动运行,随后 MSC 根据调度策略生成执行体池并连同相关策略下发给 MSA, MSA 打开服务接口等待用户请求。

3) 服务阶段: MSP 收到用户业务请求, MSA 向执行体池分发请求,执行体池各执行体处理用户请求并分别向 MSA 返回处理结果, MSA 根据裁决策略进行输出裁决,裁决的一致结果向用户输出。当裁决有不一致的情况时,将占少数的结果对应的执行体节点信息反馈给 MSC,由 MSC 根据策略进行相关操作,若有必要则进行执行体池动态调度,更换执行体池并把新的执行体池下发给 MSA,并通知云平台把有异常的执行体下线进行清洗和恢复。图 5 给出了一次请求处理过程的时序图。

由图 5 可知,在 3 余度条件下(其他余度情况相同),用户的请求将由 MSA 分发给 3 个执行体 VN 进行处理,而执行体处理该请求的耗时是不一样的,向 MSA 响应的时延就会有不一致, MSA 必须等到时延最长的响应到达后才能进行裁决操作,这在一定限度上造成拟态云服务的响应时延比非拟态云服务大,再加上一致性裁决需要消耗处理时间,时延会进一步加大,因此需要考虑减小响应时延的问题。

4.2 执行体调度机制

执行体调度是 MSP 保持其执行体池动态性、

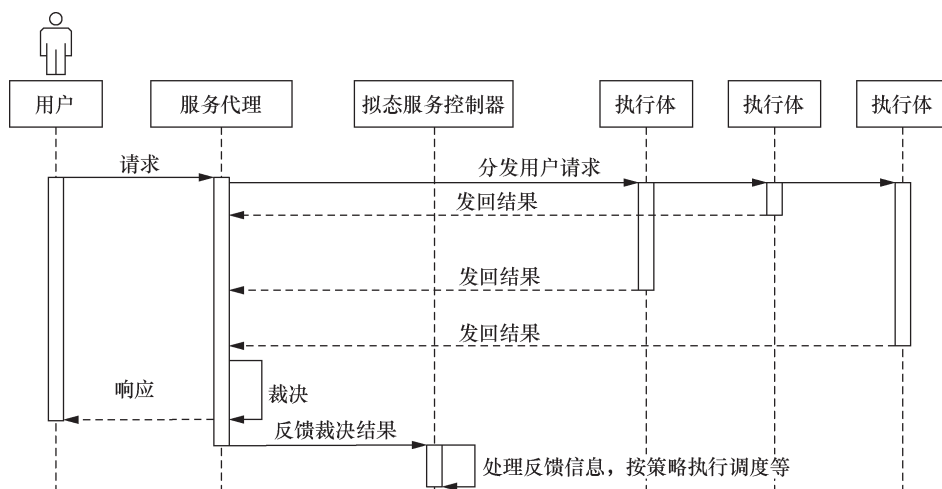


图 5 请求处理过程时序图

Figure 5 Request processing sequence

异构性、冗余性的关键机制，有以下 3 种情况触发调度。

(1) 裁决触发

MSA 执行输出裁决后，若发现有少数不一致的输出，则把对应执行体信息反馈给 MSC，MSC 根据相关策略触发调度，把该执行体换出当前执行体池，并按策略调入另一个执行体组成新的执行体池，对于 MSP 来说，就是换了一张当前执行体池配置表。MSC 还会通知云平台把换出的执行体进行清洗或者删除重建，该执行体恢复后将重新上线进入异构资源池待用。

(2) 人工触发

管理者根据实际情况调整执行体池的冗余度参数时，MSC 需要根据新的冗余度参数生成新的执行体池列表，并启动调度过程更换当前正在使用的执行体池。更换的一种方式是根据实际情况选择立即更换异构性最好的执行体池以获得较好的安全性，然而这种方式可能会因换出的执行体数量较多而产生服务抖动或中断；另一种方式是更换执行体变化最少的执行体池以保障服务的平稳，但可能会造成异构性较差。此外，管理者也可以直接启动调度来更换当前执行体池，可以人为指定新的执行体池，也可由 MSC 按策略选择。

(3) 策略触发

MSC 按照相关策略触发执行体的调度，生成新的执行体池并下发给 MSA。例如，时间片策略的时间分片结束时启动调度过程，随机冗余度策略

随机更换执行体池冗余度时启动调度过程，监控策略发现执行体异常或无响应启动调度过程。

MSP 通过上述调度机制可以获得如下收益。

(1) 增加攻击难度

调度机制使 MSP 具有动态变换的特性，增加了 MSP 的不确定性，执行体的漏洞和后门难以利用，使同种攻击连续成功的概率大幅降低，难以形成完整而持续的攻击链^[26,34]。

(2) 降低拟态逃逸概率

调度机制使 MSP 的执行体池具有较好的异构性和冗余度，增加了 MSP 的鲁棒性，减小了执行体池中各执行体存在共同漏洞和后门的概率，有效降低了拟态逃逸的可能性。

(3) 平衡安全与成本

调度机制使 MSP 的动态和异构特性实现平衡互补^[31]，当动态性弱时增加异构性，当异构性差时增加动态性弥补，保障了 MSP 的安全性。执行体池的冗余度较小时，系统开销及资源占用成本较低，可通过动态性和异构性来保障 MSP 的安全；冗余度较大时，系统开销和资源占用较高，在给定异构资源池的条件下，反而损失动态性和异构性。

4.3 输出裁决机制

输出裁决是 MSP 进行拟态防御的关键环节，其能够发现执行体的异常或异样表现，进而通过裁决反馈和调度机制阻止可能的攻击行为。输出裁决的工程实现有较高复杂度，对于同一个应用

请求，异构执行体的输出呈现大多是不同的，如呈现格式、可选信息、字段命名等，不能通过简单的字符串比较来判断输出内容是否一致，有时需要采用语义分析^[35]或者抽样比对^[36]等方式进行判决，通常需要针对不同的应用采取不同的判决方法。另外，相比单执行体的直接输出，裁决机制会增加 MSP 的响应时延。

对于面向应用的云服务，包括 Web 服务和微服务等，通常采用 RESTful^[37]架构风格的 API 接口，其接口的字段定义比较明确。MSP 针对这种类型的应用服务引入响应接口模型，该模型有两个要素（数据格式和数据字段），其中，数据格式通常是 JSON、XML 或者自定义。模型要求执行体向 MSA 返回数据时必须满足这两个要素，即要求加入 MSP 的执行体按照规定的格式和字段向 MSA 返回数据。接口模型的配置如表 2 所示。

表 2 接口模型配置
Table 2 Interface model configuration table

配置项	描述
API_ID	API 识别 ID 号
PATTERN	数据格式（JSON、XML、自定义）
FIELDS_NUMBER	API 返回字段数量
FIELDS	字段列表

管理者开发应用服务的多个执行体时，按照接口模型的配置要求向 MSP 提供对应的配置表，告知 MSA 执行体返回的数据格式和字段列表，MSA 将其记录为一个配置表。通过使用接口模型，可以大幅简化裁决机制的实现，提高裁决效率和准确性。

裁决机制如下：MSP 首先利用配置表指明的数据格式和数据字段对各执行体的处理结果进行一致性比对，比对过程要求格式和字段都一致；通过结果比对将各执行体按输出内容进行分组，然后对各分组进行大数裁决选出占多数的分组作为唯一输出。

在实际应用中若出现不能选出唯一多数分组的情况，则称大数裁决失效。此时，可依据各执行体在以往多次裁决过程中的胜出次数，直接选择胜出次数最多的执行体作为唯一输出。

5 实验分析

本节采用仿真的方法对 MSP 响应时延和安全性进行测试分析。设备配置为：Intel Core i7 1.80 GHz CPU 和 16 GB RAM，软件系统采用 Ubuntu Linux 版本 18.04LTS 和 JDK 12.0.1，工具为 Java、Python、CloudSim。实验采用 3 余度执行体池开展。

5.1 响应时延测试

本文将只有一个执行体进行业务处理的系统称为单执行体系统，在实验中也称作非拟态系统，MSP 系统称为拟态系统。通常情况下，拟态系统响应时延相对非拟态系统有所增加，下面对这一情况进行深入分析。

（1）时延模型定义

为便于分析，给出能从基本上反映时延关系的相关模型。其中，单个执行体的响应时延为

$$t = \frac{l}{p} \quad (1)$$

其中， l 表示执行体需要处理的任务量， p 表示执行体的处理性能， t 为响应时延，任务量越大时延越大，性能越好时延越小。其中，影响执行体处理性能的因素很多，包括 CPU 处理性能、操作系统性能 and 应用程序设计优劣等方面。式(1)中，忽略了网络时延等其他因素。

MSP 的响应时延表示为

$$T = \max(t_1, t_2, t_3) + t_d \quad (2)$$

其中 t_1, t_2, t_3 表示单个执行体时延， t_d 表示裁决时延，MSP 的响应时延 T 是 3 个执行体时延的最大值加上裁决时延。

定义 5 时延离散度

执行体的性能差异使其响应时延有长有短，单个执行体的性能波动也会使每次请求的响应时延不同，执行体响应时延的差异程度被称为时延离散度，记为 D 。时延离散度采用标准差进行计算，如式(3)所示。

$$D = \sqrt{\frac{\sum_{i=1}^n (t_i - \bar{t})^2}{n-1}} \quad (3)$$

其中, n 表示请求次数, t_i 表示第 i 次请求的时延, $\bar{t}$ 表示时延平均值。

(2) 时延数据采集与分析

进一步分析时延离散度前, 在两个不同的虚拟机中分别仿真相同的业务处理内容, 但业务处理内容虽然一样, 处理方法却不相同, 通过这种设置来代表两个异构执行体。然后使用蒙特卡洛法^[38]分别进行 10 000 次仿真业务, 并采集这两个异构执行体业务处理的时延数据, 接着利用科学计算工具 SciPy^[39]对采集到的时延数据进行概率密度曲线拟合。时延数据概率密度拟合曲线如图 6 所示。

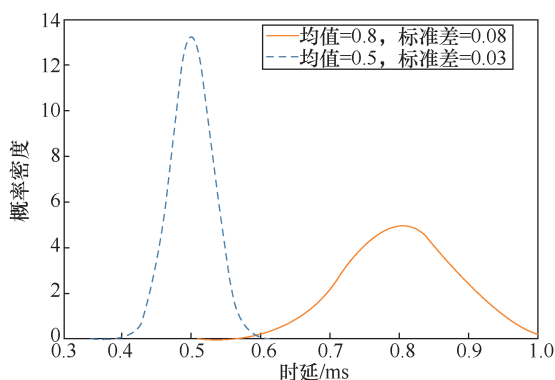


图 6 时延数据概率密度拟合曲线
Figure 6 Probability density fitting curve of time delay data

如图 6 所示, 对基于不同虚拟机及不同处理方法的两个执行体进行时延数据拟合, 分别得到均值 0.8、标准差为 0.08 的正态分布拟合曲线和均值 0.5、标准差为 0.03 的正态分布拟合曲线, 相关取值进行了近似处理。由图 6 可知, 不同的异构执行体的响应时延数据分布和正态分布拟合, 且不同异构执行体的时延分布有明显的区别。因此, 为了能够分析更多异构执行体的时延情况并简化实验条件聚焦问题分析, 本文通过构造不同的正态分布数据来仿真 MSP 执行体的时延。

(3) 时延离散度分析

通过改变正态分布的参数, 构造不同离散度的时延数据。同时, 为聚焦研究执行体时延问题, 假设裁决时延为 0。

实验 1 离散度固定, 即固定正态分布的参数为 (0.5, 0.03), 重复进行 500 次实验, 每次实验增加 10 个请求, 分别测量 MSP 和非拟态系统每次实验的请求响应时延平均值。结果如图 7 所示。

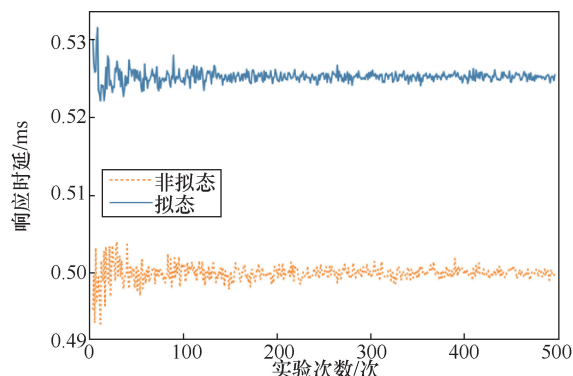


图 7 响应时延测试
Figure 7 Response delay tests

如图 7 所示, 拟态系统的响应时延整体大于非拟态系统。当请求数增大时, 时延平均值趋于稳定。分析可知, 在同样的离散度条件下, 针对每次请求, MSP 的 3 个执行体都要进行 1 次处理, MSA 必须等齐 3 个执行体的处理结果并经过裁决才能向最终用户响应, 而非拟态系统由单个执行体返回响应。由于执行体的响应时延总是波动的, 且 MSP 取执行体时延的最大值, 该值大于非拟态系统的时延值的概率较大。排除执行体的设计缺陷等异常问题, 执行体在正常情况下的性能是相对稳定的, 波动幅度不会太大, 处理的请求达到一定数量后, 其响应时延均值必然会趋于稳定, 这符合大数定律。

实验 2 请求次数固定为 10 000 次, 调整正态分布的参数使时延离散度逐渐增大, 对于每个时延离散度分别测量 MSP 和非拟态系统的平均响应时延, 结果如图 8 所示。

在图 8(a)所示的时延离散度情况下, 拟态 MSP 的平均时延总是大于非拟态的平均时延, 时延离散度越大, 两者的平均时延差越大, 而图 8(b)则更清晰地呈现了这一趋势, 即随着时延离散度的增加, MSP 的平均时延比非拟态系统越来越大。分析可知, 当时延离散度较大时, MSP 的 3 个执行体的最大时延取一个较大的值的概率较大, 使多次请求的平均值偏大; 反之, 时延离散度越小, MSP 的响应时延越接近非拟态系统的单个执行体时延。

由式(1)可知, 在任务量不变的情况下, 时延由执行体的性能决定。实际应用拟态 MSP 时, 为减小响应时延, 可以有如下优化步骤。

1) 要求执行体池中的执行体不但功能等价,

而且性能接近,以减小时延离散度。

2) 要求单个执行体的性能尽可能稳定,不产生大幅波动。

3) 提高裁决的处理性能,以减小裁决时延。

4) 对实时性要求较高的应用,可要求异构执行体性能相等,并可采用延后裁决的策略^[40],以此获得和非拟态系统接近甚至相同的响应时延。

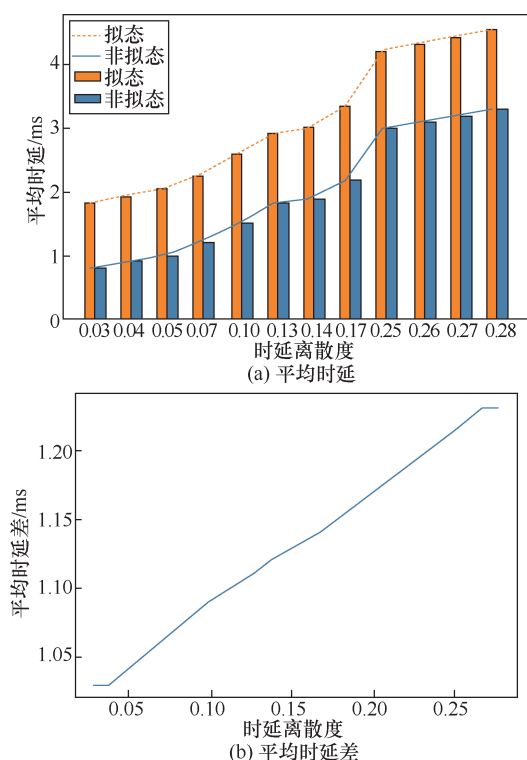


图 8 平均时延随离散度变化情况
Figure 8 The mean time delay varies with the dispersion

可以采用业界标准化的测试工具对备选执行体的性能、时延等指标进行测试分析,计算其性能或者时延离散度,从而选取合适的执行体。

此外,结合实验 1 可知,在给定离散度时,MSP 和非拟态系统的平均时延差是稳定的,由此可知,通过减小时延离散度获得的系统时延优化是稳定的。具体应用中,还需考虑执行体因异常导致的超长时延问题,可考虑设置时延阈值^[41],由 MSP 提供相应超时处理机制。

5.2 安全性测试与分析

实验使用云仿真工具 CloudSim^[42]构造 MSP 模拟 3 余度的情况进行,对比测试拟态 MSP 和非拟态单执行体的安全性。

实验设置 3 个异构执行体,其编号为 $E1$ 、 $E2$ 、

$E3$,按照共同漏洞指标的异构性要求^[31]在 $E1$ 上预置漏洞 a , $E2$ 上预置漏洞 b , $E3$ 上预置漏洞 c ,各执行体没有共同漏洞。先将这 3 个执行体作为非拟态单执行体运行测试,然后把它们组合成 MSP 的执行体池进行测试。

实验设置 3 个攻击者,编号为 $A1$ 、 $A2$ 、 $A3$,攻击者按序号分别发现执行体的漏洞,如 $A1$ 发现了 $E1$ 的漏洞 a 。实验中,观察拟态 MSP 和非拟态单执行体的防护情况。实验假设只要攻击者发现执行体的漏洞,则对该执行体的攻击必然成功。结果如表 3 所示。

表 3 安全性测试结果
Table 3 Security testing results

攻击者	$E1$	$E2$	$E3$	MSP
$A1$	✓	×	×	×
$A2$	×	✓	×	×
$A3$	×	×	✓	×

注:“✓”表示攻击成功,“×”表示攻击失败。

由表 3 的实验结果可知,在非拟态情况下, $A1$ 对 $E1$ 发起的攻击获得成功,对其他没有发现漏洞的执行体攻击失败;攻击者 $A2$ 和 $A3$ 也获得同样的效果。把 3 个执行体加入 MSP 构造拟态防御后,虽然原来的漏洞还存在,攻击者也获知了对应的漏洞,但对 MSP 的攻击还是失败了。下面以 $A1$ 为主分析实验结果,其他攻击者的情况相同。

分析可知,当把 3 个执行体构造为拟态系统 MSP 后,MSP 的裁决机制将发挥作用。按照裁决机制使用的大数判决方法,当相同输出的执行体

数量大于或等于 $\left\lceil \frac{n+1}{2} \right\rceil$ (n 表示执行体池的余

度)时,这些执行体的输出将作为 MSP 的最终输出。因此,虽然 $A1$ 对 MSP 中的 $E1$ 攻击成功,获得了 $A1$ 期望的输出;但对 $E2$ 、 $E3$ 失败, $E2$ 和 $E3$ 的输出结果一致且成为多数方。裁决机制选择了 $E2$ 和 $E3$ 的输出作为最终输出, $A1$ 未能获得期望的结果,其攻击失败。

裁决机制不但在 MSP 的输出环节终止了 $A1$ 攻击链的建立,同时发现了 $E1$ 的异常情况,其输出了和其他执行体不一致的异样数据。这时,将触发 MSP 的调度机制, $E1$ 将被调出执行体池,

换入另一个执行体,同时 MSC 会通知云平台把 $E1$ 进行清洗或者删除重建,待 $E1$ 恢复后重新上线进入异构资源池待用。因此,通过裁决机制和调度机制的协同联动, $A1$ 将很难连续重现对 $E1$ 的攻击, $A1$ 将发现 MSP 不可测、不确定,进而使 MSP 获得安全性。

然而,若 $A1$ 同时获知了 $E1$ 和 $E2$ 的漏洞信息,或者 $E1$ 和 $E2$ 存在共同漏洞,则 $A1$ 将同时从 $E1$ 和 $E2$ 获得期望的输出结果,它们的输出成为多数方,被裁决机制作为最终输出。此时, MSP 出现拟态逃逸, $A1$ 攻击成功。

下面进一步分析拟态逃逸率和余度设置相关情况。

假设执行体池中有 r ($3 \leq r \leq 21$) 个执行体,为了简化仿真测试,设各执行体被攻击成功的概率相同,概率为 P ,且 $P=0.5$;然后采用文献[26]的拟态逃逸概率计算方法,即拟态逃逸概率 $P_t = P_1 \times P_2 \times \dots \times P_r$,其中 P_1 到 P_r 分别表示第 1 号执行体到第 r 号执行体被攻击成功的概率,且它们都等于 P , r 表示执行体池的余度。仿真结果如图 9 所示。

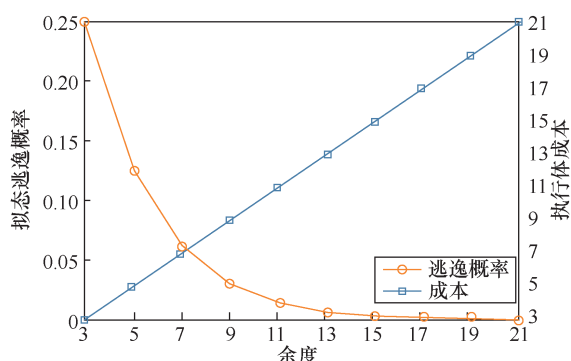


图 9 余度、成本及拟态逃逸概率关系

Figure 9 The relationship between redundancy, cost and mimic escape probability

由图 9 可知,拟态逃逸率随着余度增加而减小,并且从 3 余度到 7 余度的减小幅度最显著,之后的减小幅度不太明显;其次,随着余度增加, MSP 的执行体成本呈线性快速增长。分析可知,增加执行体池余度可以减小 MSP 发生拟态逃逸的概率,进而增加系统的安全性;但余度增加超过一定数量时,给 MSP 带来的安全增益将快速减弱,反而会大幅增加系统成本,增加系统时延、系统复杂度等。因此,结合文献[26,30]的研究,

MSP 通常采用余度为 3 的执行体池,3 余度能获得最佳性价比,在需要保持足够的动态性和可靠性时可通过调度机制适当增加冗余度。

综上所述, MSP 通过调度机制和裁决机制使系统保持动态异构冗余特性,并对攻击者呈现不可测和不确定性,使攻击者难以维持有效攻击链,进而获得安全性,其防御机理如图 10 所示。

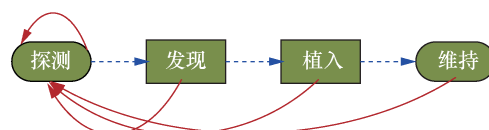


图 10 攻击链及 MSP 防御机理

Figure 10 Attack chain and MSP defense mechanism

如图 10 中蓝色箭头所示,攻击者对系统进行攻击通常会经过探测、发现、植入、维持等过程^[26,34],通过这一过程形成一个完整的攻击链,且攻击链中每个步骤的操作都依赖上一步的成功实现。然而,如红色箭头所示, MSP 通过动态调度可能会在其中的某个步骤中断攻击链,使攻击失败。这时,攻击者需要重新建立攻击链,而 MSP 的动态异构冗余特性增加了攻击者重新建链的周期和难度,输出裁决机制使攻击者难以探测和发现期望的目标。对于离散的拟态逃逸事件, MSP 通过调度机制不断变换执行体池,使同一问题的拟态逃逸无法连续发生,攻击者因此无法维持攻击链,进而获得攻击阻断,实现 MSP 的安全防护。

6 结束语

面对未知漏洞和后门带来的安全威胁,研究人员从基础硬件层、网络服务层、数据存储层、虚拟化层等多个切面^[43]取得拟态化研究进展。本文面向应用服务层提出构建云服务节点的拟态化架构,使云服务具备内生安全性和鲁棒性,可作为云平台整体拟态化解决方案的补充。

当然,拟态云服务架构的诸多方面还需要完善,下一步主要从 MSP 多形态部署、混合部署、跨云部署着手,进而研究跨云部署带来的网络时延与执行体调度问题。

参考文献:

- [1] GIRMA A, GARUBA M, LI J. Analysis of security vulnerabilities of cloud computing environment service models and its main

- characteristics[C]//2015 12th International Conference on Information Technology-New Generations. 2015: 206-211.
- [2] CHOU T S. Security threats on cloud computing vulnerabilities[J]. International Journal of Computer Science and Information Technology, 2013, 5: 79-88.
- [3] DARWISH M, OUDA A, CAPRETZ L F. Cloud-based DDoS attacks and defenses[C]//International Conference on Information Society (i-Society 2013). 2013: 67-71.
- [4] BARAKA H B, TIANFIELD H. Intrusion detection system for cloud environment[C]//Proceedings of the 7th International Conference on Security of Information and Networks. 2014: 399-404.
- [5] AL-SALEH M I, HAMDAN H M. On studying the antivirus behavior on kernel activities[C]//Proceedings of the 2018 International Conference on Internet and E-Business. 2018: 158-161.
- [6] MAVROMOUSTAKOS S, PATEL A, CHAUDHARY K, et al. Causes and prevention of SQL injection attacks in web applications[C]//Proceedings of the 4th International Conference on Information and Network Security. 2016: 55-59.
- [7] OUSMANE S B, MBACKE B C S, IBRAHIMA N. A game theoretic approach for virtual machine allocation security in cloud computing[C]//Proceedings of the 2nd International Conference on Networking, Information Systems & Security. 2019: 1-6.
- [8] ALNAIM A, ALWAKEEL A, FERNANDEZ E B. A misuse pattern for compromising VMs via virtual machine escape in NFV[C]//Proceedings of the 14th International Conference on Availability, Reliability and Security. 2019: 1-6.
- [9] LINDEMANN J. Towards abuse detection and prevention in IaaS cloud computing[C]//2015 10th International Conference on Availability, Reliability and Security. 2015: 211-217.
- [10] YANG C, GUO Y F, HU H C, et al. An effective and scalable VM migration strategy to mitigate cross-VM side-channel attacks in cloud[J]. China Communications, 2019, 16(4): 151-171.
- [11] ZHANG Y, JUELS A, REITER M K, et al. Cross-tenant side-channel attacks in PaaS clouds[C]//Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. 2014: 990-1003.
- [12] YOUNIS Y A, KIFAYAT K, HUSSAIN A. Preventing and detecting cache side-channel attacks in cloud computing[C]//Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing. 2017: 1-8.
- [13] Common vulnerabilities and exposures[EB]. 2015.
- [14] ANAND V. Intrusion detection: tools, techniques and strategies[C]//Proceedings of the 42nd Annual ACM SIGUCCS Conference on User Services. 2014: 69-73.
- [15] 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. 2014.
- [16] ZHOU X, LU Y, WANG Y, et al. Overview on moving target network defense[C]//2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC). 2018: 821-827.
- [17] WANG S, ZHANG L, TANG C. A new dynamic address solution for moving target defense[C]//2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference. 2016: 1149-1152.
- [18] ZHANG J Z, FENG X W, WANG D X, et al. Web service applying moving target defense[C]//2018 IEEE Third International Conference on Data Science in Cyberspace (DSC). 2018: 640-645.
- [19] 郭江兴. 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.
- [20] 张铮, 刘浩, 谭力波, 等. 工控拟态安全处理器验证系统测试及安全分析[J]. 通信学报, 2018, 39(S2): 131-137.
- ZHANG Z, LIU H, TAN L B, et al. Industrial control mimic security processor verification system test and security analysis[J]. Journal on Communications, 2018, 39(S2): 131-137.
- [21] ZHENG J, WU G, WEN B, et al. Research on SDN-based mimic server defense technology[C]//Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science. 2019: 163-169.
- [22] 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.
- [23] 顾泽宇, 张兴明, 林森杰. 基于拟态防御理论的 SDN 控制层安全机制研究[J]. 计算机应用研究, 2018, 35 (7): 2148-2152.
- GU Z Y, ZHANG X M, LIN S J. Research on security mechanism for SDN control layer based on mimic defense theory[J]. Application Research of Computers, 2018, 35(7): 2148-2152.
- [24] 王祺鹏, 扈红超, 程国振. 一种基于拟态安全防御的 DNS 框架设计[J]. 电子学报, 2017, 45(11): 139-148.
- WANG Z P, HU H C, CHENG G Z. A DNS architecture based on mimic security defense[J]. Acta Electronica Sinica, 2017, 45(11): 139-148.
- [25] 陈越, 王龙江, 严新成, 等. 基于再生码的拟态数据存储方案[J]. 通信学报, 2018, 39(4): 21-34.
- CHEN Y, WANG L J, YAN X C, et al. Mimic storage scheme based on regenerated code[J]. Journal on Communications, 2018, 39(4): 21-34.
- [26] 全青, 张铮, 张为华, 等. 拟态防御 Web 服务器设计与实现[J]. 软件学报, 2017, 28(4): 883-897.
- TONG Q, ZHANG Z, ZHANG W H, et al. Design and implementation of mimic defense Web server[J]. Journal of Software, 2017, 28(4): 883-897.
- [27] 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.
- [28] GARCIA M, BESSANI A, GASHI I, et al. Analysis of operating system diversity for intrusion tolerance[J]. Software-practice & Experience, 2014, 44(6): 735-770.
- [29] JOHNSTON W. Increasing system reliability-a survey of redundant control methods[C]//Fourth Annual Canadian Conference Proceedings., Programmable Control and Automation Technology Conference and Exhibition. 1988.
- [30] 潘计辉, 张盛兵, 张小林, 等. 三余度机载计算机设计与实现[J]. 西北工业大学学报, 2013, 31(5): 798-802.

- PANG J H, ZHANG S B, ZHANG X L, et al. Design and realization of treble-redundancy management method of flight control system[J]. Journal of Northwestern Polytechnical University, 2013, 31(5): 798-802.
- [31] 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.
- [32] 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. The executors scheduling algorithm for the Web server with mimic construction[J]. Computer Engineering, 2019, 45(8): 14-21.
- [33] 武兆琪, 张帆, 郭威, 等. 一种基于执行体异构度的拟态裁决优化方法[J]. 计算机工程, 2019, 10: 1-8.
- WU Z Q, ZHANG F, GUO W, et al. A mimic ruling optimization method based on executive heterogeneity[J]. Computer Engineering, 2019, 10: 1-8.
- [34] 刘文彦, 霍树民, 陈扬, 等. 网络攻击链模型分析及研究[J]. 通信学报, 2018, 39(S2): 88-94.
- LIU W Y, HUO S M, CHEN Y, et al. Analysis and study of cyber attack chain model[J]. Journal on Communications, 2018, 39(S2): 88-94.
- [35] QI Q, WU J X, HU H C, et al. An intensive security architecture with multi-controller for SDN[C]//2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). 2016: 401-402.
- [36] 李文彬, 刘璇, 张建畅, 等. 基于随机抽样一致性的误匹配剔除方法研究[J]. 计算机仿真, 2019, 36(10): 233-237.
- LI W B, LIU X, ZHANG J C, et al. Mismatching culling algorithm based on minimum distance and RANSAC fusion[J]. Computer Simulation, 2019, 36(10): 233-237.
- [37] CHRISTENSEN J H. Using RESTful Web-services and cloud computing to create next generation mobile applications[C]//Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications. 2009: 627-634.
- [38] 巴斌, 郑娜娥, 朱世磊, 等. 利用蒙特卡洛的最大似然时延估计算法[J]. 西安交通大学学报, 2015, 49(8): 24-30.
- BA B, ZHENG N E, ZHU S L, et al. A maximum likelihood time delay estimation algorithm using monte carlo method[J]. Journal of Xi'an JiaoTong University, 2015, 49(8): 24-30.
- [39] VIRTANEN P, GOMMERS R, OLIPHANT T E, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python[J]. Nature Methods, 2020, 17(3): 261-272.
- [40] WANG Y W, WU J X, GUO Y F, et al. Scientific workflow execution system based on mimic defense in the cloud environment[J]. Frontiers of Information Technology & Electronic Engineering, 2018, 19(12): 1522-1537.
- [41] 聂德雷, 赵博, 王崇, 等. 拟态多执行体架构下的超时阈值计算方法[J]. 网络与信息安全学报, 2018, 4(10): 68-76.
- NIE D L, ZHAO B, WANG C, et al. Timeout threshold estimation algorithm in mimic multiple executors architecture[J]. Chinese Journal of Network and Information Security, 2018, 4(10): 68-76.
- [42] CALHEIROS R N, RANJAN R, BELOGLAZOV A, et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms[J]. Software: Practice and Experience, 2011, 41(1): 23-50.
- [43] 周清雷, 冯峰, 朱维军. 基于功能切片的拟态防御体系结构及安全等级评估方法[J]. 通信学报, 2018, 39(S2): 95-105.
- ZHOU Q L, FENG F, ZHU W J. Mimic defense organization structure based on functional slice and method of evaluating security level[J]. Journal on Communications, 2018, 39(S2): 95-105.

[作者简介]



普黎明 (1976-), 男, 云南嵩明人, 信息工程大学副研究员, 主要研究方向为网络安全、网络体系结构。



卫红权 (1971-), 男, 河南唐河人, 博士, 信息工程大学研究员, 主要研究方向为融合网络安全、可重构网络理论与技术。



李星 (1987-), 男, 河南新乡人, 博士, 信息工程大学助理研究员, 主要研究方向为链路预测、社团挖掘。



江逸茗 (1984-), 男, 江苏南通人, 博士, 信息工程大学讲师, 主要研究方向为网络虚拟化、网络架构。