一种基于拟态防御的差异化反馈调度判决算法

高明，罗锦，周慧颖，焦海，应丽莉
（浙江工商大学信息与电子工程学院，浙江 杭州 310018）

摘　要：面对服务路径的安全性问题，根据拟态防御理论里的基于动态异构冗余（dynamic heterogeneous redundancy，DHR）模型的服务功能链部署拟态防御体系架构，并结合服务路径部署的实际需求，提出了一种基于拟态防御的差异化反馈调度判决算法。首先依据调度算法中执行体集的异构度及安全防御系数，从执行体池中筛选出适合拟态防御场景的调度器，然后根据判决算法的可靠度系数及多数判决算法选出判决器，最后，对本文算法与普通的调度算法、判决算法进行仿真分析。仿真结果表明，本文算法可有效地提升系统的防御能力，保障服务路径配置的安全。

关键词：拟态防御；调度算法；判决算法；网络安全
中图分类号：TP393
文献标识码：A
doi: 10.11959/j.issn.1000–0801.2020146

A differential feedback scheduling decision algorithm based on mimic defense

GAO Ming, LUO Jin, ZHOU Huiying, JIAO Hai, YING Lili
School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China

Abstract: Facing the security problem of the service path, according to the service function chain deployment mimic defense architecture based on dynamic heterogeneous redundancy (DHR) model in the mimic defense theory, and combined with the actual needs of the service path deployment, a differential feedback scheduling decision algorithm based on mimic defense was proposed. Firstly, according to the heterogeneity of the executive set in the scheduling algorithm and the security defense coefficient, the scheduler suitable for the mimic defense scenario was selected from the executive pool, and then the decider was selected according to the reliability coefficient of the decision algorithm and the majority decision algorithm. The proposed algorithm, common scheduling algorithm and decision algo-

收稿日期：2020–01–08；修回日期：2020–04–26
通信作者：高明，497862918@qq.com
基金项目：国家重点研发计划基金资助项目（No.2017YFB0803202）；国家自然科学基金资助项目（No.61871468）；浙江省自然科学基金资助项目（No.LY18F010006）；浙江省新型网络标准与应用技术重点实验室基金资助项目（No.2013E10012）；浙江省重点研发计划基金资助项目（No.2019C01056，No.2020C01079）
Foundation Items: The National Key Research and Development Program of China (No.2017YFB0803202), The Natural Science Foundation of China (No.61871468), Zhejiang Provincial Natural Science Foundation of China (No.LY18F010006), Zhejiang Provincial Key Laboratory of New Network Standards and Technologies (NNST) (No.2013E10012), Zhejiang Provincial Key Research and Development Program (No.2019C01056, No.2020C01079)

rithm were simulated and analyzed. Simulation results show that the proposed algorithm can effectively improve the system's defense capabilities and ensure the security of the service path configuration.

Key words: mimic defense, scheduling algorithm, decision algorithm, network security

1 引言

传统网络利用服务链让数据流量经过各网络节点，但由于其效率低下、配置复杂、设备成本昂贵等，业界开始采用软件定义网络（software defined networking，SDN）[1]。SDN 的服务路径配置是指 SDN 控制器通过 OpenFlow 协议[2]下发流表，控制数据流的转发路径，对数据流进行快速配置[3]，规定其经过的网络节点。但由于 SDN 控制器缺乏相关的有效安全算法，攻击者很容易攻击控制器，然后对网络设备的转发行为进行更改。为了保障 SDN 里服务路径配置的安全，需要保证 SDN 控制器的安全以及 SDN 控制器对交换机路径的正确配置[4]。业界为解决 SDN 控制器安全问题进行了很多讨论，并在解决问题的思路上分成了两个方向：一种是为已有的 SDN 控制器进行安全功能扩展，设计出了 FortNOX[5]、SE-Floodlight[6]、FRESCO[7]、OFX[8]等控制器；另一种是设计全新的、具有安全算法的 SDN 控制器。目前新的 SDN 控制器，如 Floodlight、OpenDaylight 等对安全越来越重视，但仍然不可避免地出现漏洞，表 1 列出了某些主流控制器的安全漏洞。

表 1 某些主流控制器的安全漏洞

控制器	代码量	已知漏洞
Floodlight	约 18 万行	拓扑欺骗漏洞
OpenDaylight	约 80 万行	XXE 漏洞、Netdump 漏洞
ONOS 1.11.0 版	约 110 万行	Root 缺陷漏洞、DoS 漏洞
ONIX	非开源控制器，未知	KYE 漏洞

这些漏洞虽然可以通过后期的工作不断修复，但因为无法避免未知以及不可预测的安全威胁，所以并不能从根本上解决 SDN 控制器面临的安全性问题。为提升网络空间的安全性，鉴于目前网络架构的静态性、单一性及现有防御技术无法对未知风险形成有效的防御，部分研究人员提出了拟态防御技术。该技术利用异构性、动态性、多样性改变了现有防御系统的脆弱性、静态性、单一性，提高网络空间的防御能力，并改变了目前网络空间易攻难守的局面。本文根据拟态防御理论里的基于动态异构冗余（dynamic heterogeneous redundancy，DHR）模型的服务功能链部署拟态防御体系架构[9]，并结合服务路径部署的实际需求，提出了一种基于拟态防御的差异化反馈调度判决算法，通过动态变换异构调度器和判决器的算法以有效主动防御攻击者的入侵，提高 SDN 的安全性。

2 相关研究现状

目前传统防御技术有很多成果，如采用静态防御技术的入侵检测、漏洞扫描[10]、防火墙等。蜜罐技术[11]不同于静态防御技术，它类似于一种利用网络欺骗攻击者的安全防御技术，首先使网络攻击者消耗攻击资源，再获取其攻击信息以加强其自身的安全防御。但是就目前形势来看，传统静态防御技术已满足不了网络安全的需求，这些入侵容忍技术无法很好地保障网络的安全性。

在新型防御技术上，网络空间的拟态防御具有颠覆性的技术优势，并且大大改变了现在网络空间易攻难守的局面，让目标防御对象获得攻击者测不准的特性，很好地防御了网络空间的未知威胁。目前已有一些初步的研究成果，拟态路由器[12]基于拟态防御技术，在路由器体系架构上引入异构冗余功能执行体，通过动态调度机制与多数判决算法，实现路由系统的主动防御；拟态 Web 服务器[13]能在较小开销的前提下防御测试中的全部攻击类型；拟态 DNS 架构部署简单并具有入侵容忍能

力。实验证明，与传统架构相比，拟态防御技术可以更好地保障系统安全。

3　基于拟态防御的调度判决算法的改进

3.1　现有算法的分析

调度算法是实现拟态防御的核心技术之一，合理的调度算法可提升系统的防御能力。调度算法的思想就是动态改变系统结构，给攻击者不确定的场景，使其无法预知改变，从而实现安全防御。目前有多种调度算法，如轮询调度算法通过其周期性变化来防御攻击者的攻击，但攻击者可通过长期探测发现规律，预知其轮询变化；如随机调度算法[14]通过随机选择来防御攻击者的攻击，但对安全性能的提升有限且不便于管理。

判决算法是对多个异构执行体的多个输出结果进行比较得到一个最终输出结果，有随机判决、轮询判决、多数判决等。目前在拟态防御上主要采用多数判决算法。多数判决算法的原则是以占多数的输出结果为先，认为该结果是安全可靠的并作为最终结果输出。例如要配置交换机的流表信息，配置信息下发到 5 个控制器进行处理，然后控制器处理后输出 5 条流表。其中 2 台控制器受到攻击，没有输出正确的流表，判决器对 3 个输出进行多数判决后，选取另外 2 台输出结果相同的流表作为最终输出结果，拟态的多数判决成功避免了错误流表的下发。

现有的许多调度判决算法并不是基于拟态防御的应用场景设计的，适用上有许多不足。

3.2　差异化反馈调度判决算法

3.2.1　调度对象和调度个数

现有的调度算法，如轮询调度算法、随机调度算法等不能很好地起到防御入侵的效果。本文对现有调度算法进行改进，设计了一种差异化反馈调度算法，调度算法主要包括两个方面：调度对象和调度个数。

调度对象为拟态控制层中的异构执行体，调度选中执行体池中若干个执行体用于执行任务，执行体间相互独立且对等。调度任务是动态选择用于执行任务的执行体，并且执行体的个数 $n \geqslant 3$。调度模型中的数学符号说明见表 2。

表 2　调度模型中的数学符号说明

数学符号	含义
E_p	执行体池中所有执行体集合
E_m	选中执行的执行体集
E_i	第 i 个执行体
status	执行体状态
σ	执行体的异构度
σ^*	执行体集的异构度
μ	安全防御系数
$\theta(\ )$	调度对象函数
U	执行体输出结果分类集合
$m(\ )$	调度个数函数

执行体池中总共有 n 个执行体 $E_p=\{E_1,E_2,E_3,\cdots,E_n\}$，$E$ 代表单个执行体，调度选择 m 个执行体组成执行体集，k 个执行体未被选中，其中 $n=m+k$。

每个执行体有 3 种状态，通过 status 来表示：status=1 表示执行体被选中；status=0 表示未被选中但之后可能被选中；status=2 表示执行体安全防御系数过低，之后不会被选中。

在拟态防御中，调度选择到的执行体对象结构差异越大，越能增加系统内部结构的复杂性，越有利于增加攻击者的攻击难度，是一种有效的安全防御手段。为了量化执行体结构的差异性，本文利用软件相似度度量（measure of software similarity，MOSS）[15-17]得到执行体之间的异构度 σ。异构度包括许多方面，如编程语言、运行的操作系统平台、硬件架构等。异构度是用于描述异构程度的一种量化参数，数值越小代表结构差异越大，且 $\sigma \in (0,1]$。通过 MOSS 计算了几种操作系统的异构度，为：

$$M\{\text{centos},\text{fedora},\text{debian},\text{centoo}\}=\begin{bmatrix}1.0 & 0.664 & 0.807 & 0.697\\ 0.664 & 1.0 & 0.593 & 0.866\\ 0.807 & 0.593 & 0.1 & 0.620\\ 0.697 & 0.866 & 0.620 & 1.0\end{bmatrix} \quad (1)$$

选择异构度大的执行体有利于增加攻击者的难度，在调度过程中对执行体设立异构度作为调度的依据。本文定义了执行体集的异构度，为：

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

其中，$\sigma(E_i,E_j)$ 为 E_i、E_j 的异构度。

执行体本身设计结构的差异会造成防御能力上的不同，对于调度器而言，应尽可能选择安全防御能力较强的执行体，不能随机选择执行体。本文根据执行体输出的异常状况定义了执行体的安全防御系数 μ，为：

$$\mu(E_P)=\{\mu_1,\mu_2,\mu_3,\cdots,\mu_n\} \quad (3)$$

初始化状态时，每个执行体的安全防御系数都是相等的，通过判决器反馈执行体是否为异常输出，根据执行体每次的输出情况，更新自身的安全防御系数，其中数值越小越安全。式（4）为安全防御系数更新公式，ϑ 为更新因子。

$$\mu=\begin{cases}\mu_{t-1}\times\vartheta, & \text{执行体输出正常}\\ \mu_{t-1}/\vartheta, & \text{执行体输出异常}\end{cases} \quad (4)$$

执行体集的安全防御系数如下：

$$\mu^{*}=\frac{1}{m}\times\sum_{i}^{m}\mu_i \quad (5)$$

其中，μ 的数值越小，代表安全性越高，且 μ、$\mu^{*}\in(0,1]$，调度器会选择 μ^{*} 值小的执行体集。

根据上述分析，综合考虑执行体的异构度、安全防御系数对安全防御的影响，定义执行体的调度对象函数如下：

$$\theta(E_p)=\frac{1}{2m}\times\sum_{i}^{m}\sum_{j}^{m}\sigma(E_i,E_j)+\frac{1}{m}\times\sum_{i}^{m}\mu_i=\sigma^{*}+\mu^{*} \quad (6)$$

σ^{*}、μ^{*} 的值越小，防御性能越好，调度算法求解调度函数的最小值，调度器选择对应的执行体，求解计算式如下：

$$\theta(E_p)=\arg\min(\sigma^{*}+\mu^{*}) \quad (7)$$

得到执行体集 $E_m=\{E_1,E_2,E_3,\cdots,E_m\}$，作为最后的调度对象。

拟态防御采用多数正确算法，对每个执行体的输出进行判决，将相同个数最多的结果作为最终的输出结果。攻击者需要攻击执行体集控制系统，执行体越多，攻击者需要攻击越多的执行体才能改变输出占比，影响判决结果。所以调度执行体的个数直接影响到判决的结果，最终影响系统的安全性。调度选择越多的执行体固然越安全，但付出的代价也越大。综合考虑得到一个较为合理的调度个数，本文采用基于判决反馈的调度数量算法，根据判决结果的输出的占比情况，更新每次调度执行体的个数。

通过判决反馈模块可以得到上一次执行体输出的占比情况，本文采用的是执行体输出的可靠度占比为判决依据，具体可见第 3.2.3 节。根据反馈模块，将相同输出归为一类后可得到各类输出的占比情况，排序后得到各类结果的占用百分比，U 为结果占比合集，为：

$$U=\{U_1,U_2,U_3,\cdots,U_k\} \quad (8)$$

其中，U_1 为最大占比，即它所对应的执行体的输出为判决器的最终输出。

U_1 体现各执行体输出结果的一致性：U_1 接近100%说明各个执行体的输出非常一致，系统安全状态也较为良好；当 U_1 变小时，说明各执行体输出较为不一致，系统内部不稳定，部分执行体可能遭受到攻击，改变了正确输出结果，系统安全状态较差。

增加执行体有利于提升系统安全性，攻击者需攻击更多执行体来改变输出占比，增加攻击者难度。权衡系统代价与安全性，当输出与上次相比较为一致时减少调度器调度个数，反之增加。

根据U_1前后时刻的变化率更新调度个数，调度个数计算式如下：

$$m(t)=\left\lceil\left(1+\alpha\left(U_1(t-2)-U_1(t-1)\right)\right)\times m(t-1)\right\rceil \quad (9)$$

其中，$m(t-1)$表示上一时刻的调度个数，α为常数，式（9）取整后得到t时刻的调度个数$m(t)$。

根据执行体输出的占比的前后变化动态地更新调度器调度执行体的个数，使系统内部结构进一步地实现动态变换，增加系统内生安全性。

3.2.2　调度算法实现过程

调度算法伪代码如下所示。

输入　执行体池E_p，n个执行体，调度个数m

输出　调度执行体集E_m

for (i=0；i<n；i++)

　if (E_i.status = = 2)

　　E_p.pop(E_i)；

　　E_a = getalllist(E_p, m)

　　$E_m = E_a[0]$；

　　for (i=0；i< len(E_p)；i++)

　　if ($\theta(E_m) > \theta(E_a[i])$)

　　$E_m = E_a[i]$；

　return　E_m

3.2.3　依据可靠度参数的多数判决算法

目前的多数判决不依赖其他数据，仅仅通过简单的对比得到多数结果然后输出，实现简单，安全性一般。轮询判决算法因为其周期性，攻击者可通过长期探测发现规律，预知其轮询变化，不能很好地保障系统安全。随机判决算法由于其随机性，不能很好地防护系统。上面几种方法的缺陷使得判决算法有很大的提升空间。

由于每个执行体本身结构的不同，自身的安全性也不同，输出结果的可靠度也不相同，所以不应该认为完全平等。本文引入可靠度系数ω，不再是每个输出相当于1分而是根据可靠度系数，可靠度系数为 1 时输出可得 1 分，为 1.1 时可得 1.1 分。图 1 为判决过程实例。

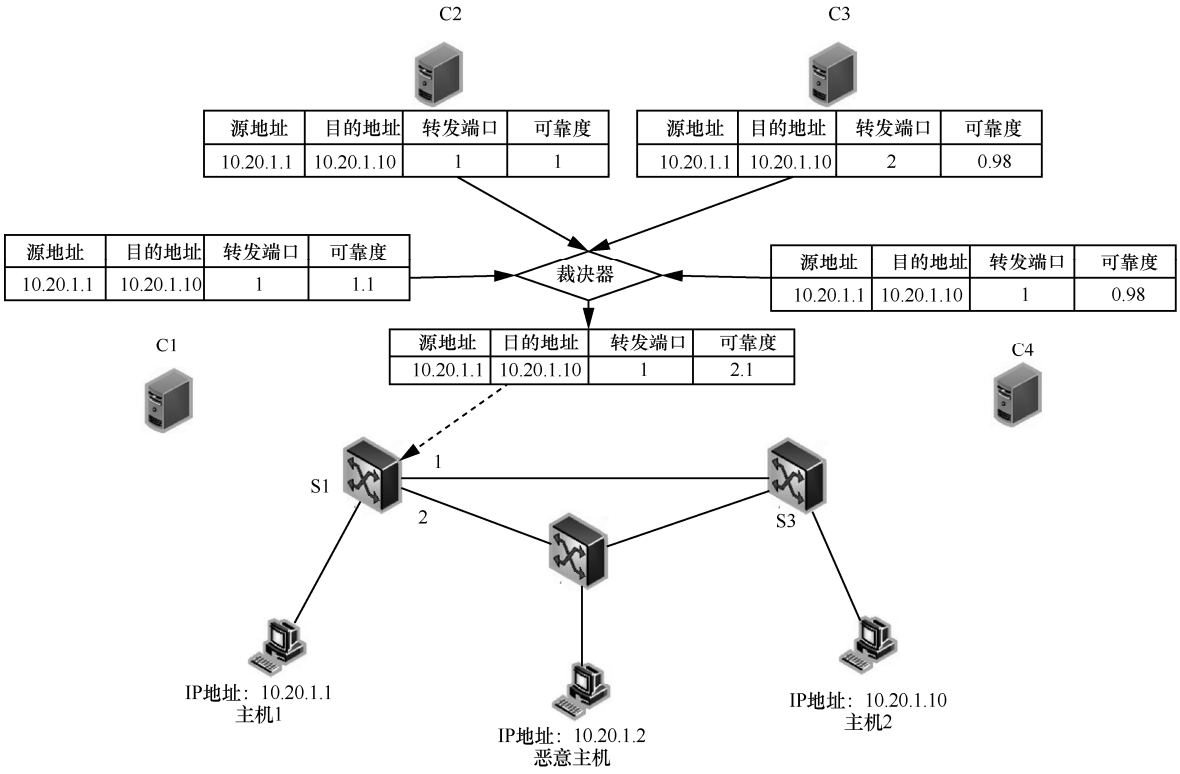

图 1　判决过程实例

图 1 中，主机 1 要发送数据给主机 2，4 个执行体生成流表下发到交换机，形成一条转发路径。攻击者想窃取数据，攻击了 C3 和 C4 执行体，使转发端口变成 2，让数据流经过 S2，通过恶意主机来窃取数据。对于 4 个执行体的输出，判决器将相同结果的归为一类，同一类的可靠度系数相加，图 1 中有两类结果，第一类转发端口为 1，第二类转发端口为 2。第一类的可靠系数之和为 2.1，第二类为 1.96，取可靠度系数最高的结果，最终判决器选择转发端口为 1 的结果作为输出，避免了数据流的错误转发。

可靠度系数 ω 根据每次判决的结果而动态变换，如一个执行体的输出一直与其他多数输出结果不同，则该执行体的可靠度系数 ω 就会降低。每个执行体初始化阶段的可靠度系数初始值 w^0 根据执行体属性中的安全防御系数设定，为：

$$w^0 = \beta \times \mu \tag{10}$$

其中，β 为折扣因子，每个执行体的可靠度系数根据判决结果更新计算式：

$$w = \begin{cases} w_{t-1} + a \times w_{t-1}^{\frac{l}{m}}, \varphi = \varphi' \\ w_{t-1} - a \times w_{t-1}^{\left(1-\frac{l}{m}\right)}, \text{其他} \end{cases} \tag{11}$$

其中，φ 表示执行体的输出，φ' 为判决器选中的最终输出，m 为当前活跃的执行体个数，l 为当前执行体输出 φ 相同的执行体个数，a 为常数弱化 w 的变化幅度。

执行体的可靠度系数更新特性为执行体输出和判决器判决结果相同 w 变大，反之变小。执行体输出与多数执行体输出不同，输出结果被判决器淘汰，当前执行体可能受到攻击，所以可靠度系数需变小。

判决器判决结果求解模型如下。

调度器选择执行体集 $E = \{E_1, E_2, \cdots, E_m\}$，执行体可靠度系数 ω 初始值 w^0 根据安全防御系数获得，执行体集可靠度系数的初始值为：

$$\{w_1^0, w_2^0, \cdots, w_m^0\} = \beta \times \{\mu_1, \mu_2, \cdots, \mu_m\} \tag{12}$$

执行体输出 m 个结果 $\{\varphi_1, \varphi_2, \cdots, \varphi_m\}$。将执行体集的输出结果相同的分为一类：

$$\varphi^* = \{[\varphi_{1,a}, \varphi_{1,b}, \cdots, \varphi_{1,c}], [\varphi_{2,a}, \varphi_{2,b}, \cdots, \varphi_{2,c}], \cdots, [\varphi_{k,a}, \varphi_{k,b}, \cdots, \varphi_{k,c}]\} \tag{13}$$

执行体集输出对应的可靠度系数 w^*：

$$w^* = f(\varphi^*) = \{[w_{1,a}, w_{1,b}, \cdots, w_{1,c}], [w_{2,a}, w_{2,b}, \cdots, w_{2,d}], \cdots, [w_{k,a}, w_{k,b}, \cdots, w_{k,e}]\} \tag{14}$$

将相同分类的 w 值相加，其中 i 代表分类，l 表示同分类的执行体个数，各类的 w 之和为：

$$w_i = w_{i,a} + w_{i,b} \ldots + w_{i,l} = \sum_{j=1}^{l} w_{i,j} \tag{15}$$

判决器最终的输出结果 φ' 为：

$$\varphi' = \arg\max F(w_i \mid \varphi) \tag{16}$$

最后输出结果为当 φ 所处分类的 w 值最大时对应的结果 $\varphi' = \varphi$。通过对执行体引入可靠度系数，增加多数判决的依据，得到最终判决结果。

4 仿真分析

4.1 仿真环境

本节对在拟态控制层设计的调度算法与判决算法进行仿真分析，测试算法的有效性。本文设计了基于 C 语言的执行体模拟器，设计了不同结构的执行体，每个执行体有不同的漏洞，每个漏洞有不同的攻击成功率，根据执行体的结构相似程度设计漏洞相同程度，执行体越相似，相同的漏洞越多。

4.2 调度判决算法组合仿真

在执行体调度中，执行体异构度越大，系统越复杂，攻击难度也越大。攻击强度函数为：

$$f(t) = \lambda e^{-\lambda t} \tag{17}$$

随着攻击时间的增加，攻击的成功率增加，为：

$$p(t) = \int f(t) dt = 1 - e^{-\lambda t} \tag{18}$$

通过式（18）可以看出，当时间无限增加时，

攻击的成功率无限接近于 1。根据前文的分析，异构程度越大，异构度数值越小，攻击的成功率越低。异构度与成功率的关系如下：

$$p_\sigma = \tau \times \sigma^* \times p_0 \tag{19}$$

其中，τ 为折扣因子，σ^* 为异构度。

设攻击者攻击正在运行的 m 个执行体，对于每个执行体的攻击是否成功是相互独立的，设它们成功攻击的概率分别为 $\{p_1, p_2, \cdots, p_m\}$，攻击者攻击 k 个执行体才能改变输出，系统被成功攻击的概率如下：

$$P_1 = C_m^k \prod_i^k p_i \prod_j^{m-k} \left(1 - p_j\right) \tag{20}$$

异构度与执行体集被成功攻击概率的关系函数为：

$$P_2 = C_m^k \prod_i^k \tau\sigma_i p_i \prod_j^{m-k} \left(1 - \tau\sigma_j p_j\right) \tag{21}$$

所以构建执行体集的异构性、增加执行体集的异构度可降低系统被成功攻击的概率，提升系统的防御能力与安全性。不同异构度下系统被成功攻击的概率如图 2 所示。

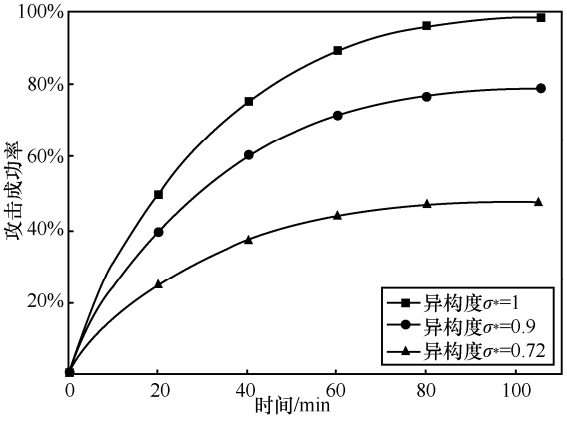

图 2　不同异构度下系统被成功攻击的概率

由图 2 可知，随着时间的增加，攻击成功率不断增加。异构度数值越低的执行体，攻击成功率也越低，增长速率也越低。实验表明，对执行体集异构度的有效选择可提升系统的防御能力。

执行体的安全防御系数根据执行体输出的异常情况动态更新，执行体输出与判决器最终输出不同则判定为异常，可反映出执行体的安全防御能力，异常次数的增多导致执行体的安全性下降。根据执行体历史发生异常数来选择调度执行体。不同调度算法的系统输出异常次数如图 3 所示。

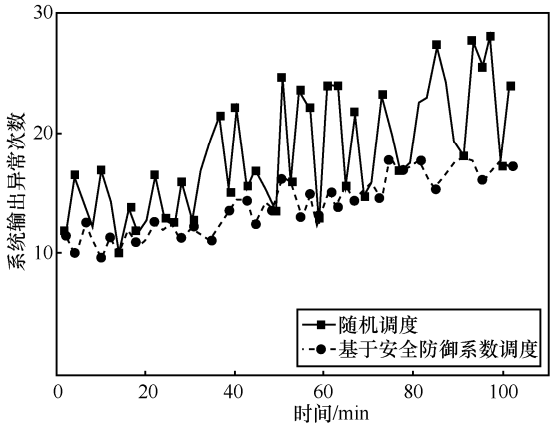

图 3　不同调度算法的系统输出异常次数

由图 3 可知，随着时间的增加，系统异常次数出现递增的趋势，表明攻击者对系统更加熟悉，攻击成功率也随之增加，随机调度的系统输出异常的平均次数为 17.77 次，而基于安全防御系数调度的系统输出异常的平均次数为 12.53 次，这是因为基于安全系数调度会选择历史异常次数相对较少的执行体，这些执行体防御能力较强。利用安全防御系数作为调度依据可有效地减少系统异常次数。

各执行体的输出结果不一致说明执行体输出异常增多、系统安全性减弱。传统的按输出结果个数的多数判决中，执行体被成功攻击的个数 k 要满足 $k>m/2$，即成功攻击个数大于执行体集中执行体个数的一半以上时，系统才被成功攻击。而本文设计的判决算法，当本文定义的可靠度系数之和大于其他执行体的可靠度系数之和时：

$$\sum_{i\in E}^{m-k} w_i < \sum_{k\in E}^{k} w_j \tag{22}$$

系统则被成功攻击，显然直接通过增加执行体集的执行体个数 m 可提升攻击者的攻击难度，

提升系统安全性。传统的调度算法的执行体调度个数是保持不变的，本文根据系统历史输出一致率状况动态地改变执行体调度个数。不同调度个数下的系统异常率如图 4 所示。不同调度个数时的系统负载率如图 5 所示。

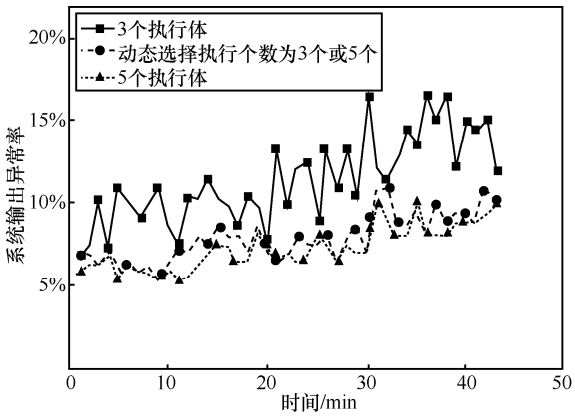

图 4　不同调度个数下的系统异常率

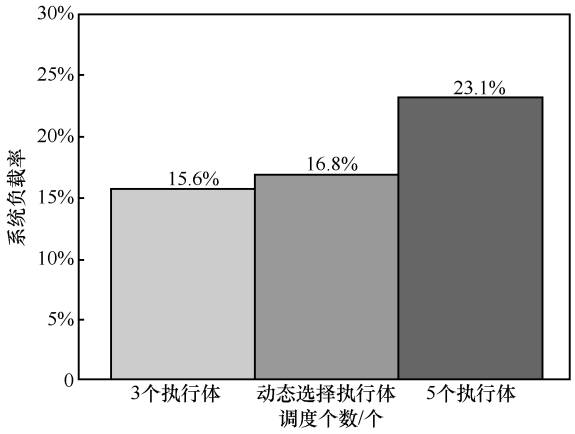

图 5　不同调度个数时的系统负载率

由图 4 可知，随着时间的增加，异常率整体呈现上升趋势。3 个执行体的系统输出异常率平均值为 0.105 8，动态选择执行体个数的系统输出异常率平均值为 0.072 1，5 个执行体的系统输出异常率平均值为 0.067 3。由图 5 可知，动态选择执行体与 3 个执行体的负载均较低，而 5 个执行体的系统负载较高。实验结果表明，根据系统历史输出一致率状况动态改变执行体调度个数既保证了较低的系统输出异常率，同时系统负载也较低，兼顾了安全性与系统代价。

执行体的结构不同，每个输出结果的可靠性也有所不同。传统的多数判决不区分执行体，无差异地对待每个执行体。本文根据执行体历史输出异常状况反馈，更新每个执行体的可靠度，以可靠度系数为判决原则，区别对待每个执行体，增加异常少即防御能力强的执行体。利用可靠度区分每个执行体的差异性，充分发挥每个执行体的作用，使防御能力进一步提升。不同判决算法下的判决出错率如图 6 所示。

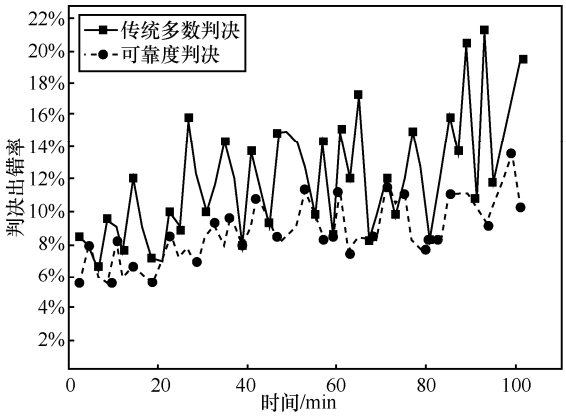

图 6　不同判决算法下的判决出错率

由图 6 可知，传统的多数判决的平均判决出错率为 0.124，可靠度判决的平均判决出错率为 0.078。传统多数判决与可靠度判决均实现了较低的判决出错率，可靠度判决结果会偏向于历史输出结果错误较少的执行体的输出，比传统多数判决实现了更低的错误率，弱化了那些防御能力较弱的执行体对判决结果的影响。实现结果表明，可靠度判决可提升系统的安全性。

不同调度算法与判决算法组合下的系统失效率如图 7 所示。由图 7 可知，随着时间的增加，系统失效率不断增加，静态方法系统的失效率急剧上升；其他两种算法由于内部结构是动态变换的，攻击者要想成功攻击需要不断地探测系统，所以系统的失效率增长缓慢。在开始阶段，三者的系统失效率较为一致，因为攻击者还没探测到系统的具体结构，后期动态调度的增长较为缓慢。差异化反

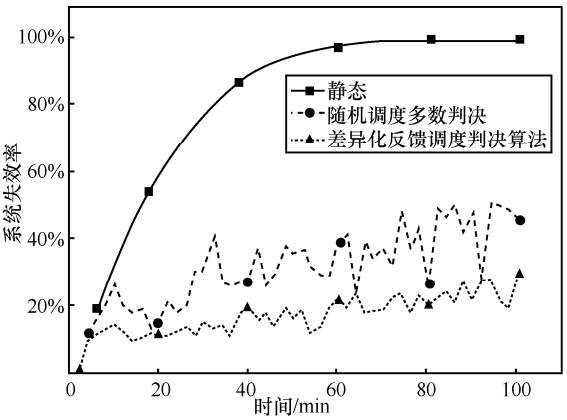

图 7　不同调度算法判决算法组合下的系统失效率

馈调度判决算法的平均失效率为 0.175，比起随机调度多数判决的 0.289，系统失效率显著地降低，这是因为基于异构度、安全防御系数、可靠度、动态调度个数、判决反馈选择最优的执行体，有效地降低了系统的失效率，而随机调度无任何依据，无法选择出最优的方案，导致失效率较高。仿真结果显示，动态调度可有效地提高系统的安全性，制定有效的调度算法和判决算法可充分发挥拟态防御的效果，从而大大增加系统的安全性。

5　结束语

拟态思想的核心是通过拟态控制层的调度判决算法进行主动防御，但是目前的调度算法和判决算法并不能很好地适应拟态防御应用场景。本文设计的差异化反馈调度判决算法通过动态变换异构调度器和判决器算法，有效主动地防御攻击者的入侵。仿真结果表明，该方法可有效地提升系统的防御能力，保障服务路径配置的安全。在后续研究中，将对该算法做相关优化，使之具有更好的实用性。

参考文献：

[1] TOURRILHES J, SHARMA P, PETTIT J, et al. SDN and OpenFlow evolution: a standards perspective[J]. Computer, 2014, 47(11): 22-29.

[2] KUŹNIAR M, PEREŠÍNI P, KOSTIĆ D, et al. Methodology, measurement and analysis of flow table update characteristics in hardware OpenFlow switches[J]. Computer Networks, 2018, 4(11): 5-16.

[3] 吴琼. 基于 SDN 的服务链的研究与实现[D]. 成都: 西南交通大学, 2017.
WU Q. Research and implementation of service chain based on SDN[D]. Chengdu: Southwest Jiaotong University, 2017.

[4] 周桥. 基于 SDNFV 的服务功能链部署优化技术研究[D]. 郑州: 信息工程大学, 2017.
ZHOU Q. Research on service function chain deployment optimization technology based on SDN[D]. Zhengzhou: Information Engineering University, 2017.

[5] PORRAS P, SHIN S, YEGNESWARAN V, et al. A security enforcement kernel for OpenFlow networks[C]//Proceedings of the HotSDN Workshop at SIG-COMM. New York: ACM Press, 2012: 121-126.

[6] CHEUNG S, FONG M, PORRAS P, et al. Securing the software-defined network control layer[Z]. 2015.

[7] KAUR R, SINGH A, SINGH S, et al. Security of software defined networks: taxonomic modeling, key components and open research area[C]//Proceedings of International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT). Piscataway: IEEE Press, 2016.

[8] SONCHACK J, AVIV A J, KELLER E, et al. POSTER: OFX: enabling OpenFlow extensions for switch-level security applications[C]//Proceedings of ACM SIGSAC Conference on Computer & Communications Security. New York: ACM Press, 2015.

[9] MIN J X. Research on mimic defense in cyberspace [J]. Journal of Information Security, 2016, 1(4): 1-10.

[10] GUO W B, LI F. Research on Web application security vulnerability scanning technology[J]. Information Communications, 2017(12): 123-124.

[11] SHI L Y, LI Y, MA M F. New progress in honeypot technology research[J]. Journal of Electronics & Information Technology, 2019, 41(2): 249-259.

[12] MA H L, YI P, JIANG Y K, et al. Mimic defense architecture of router based on dynamic heterogeneous redundant algorithm [J]. Journal of Information Security, 2017, 2(1): 29-42.

[13] 仝青, 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

[14] URGO M, VANCZA J. A branch-and-bound approach for the single machine maximum lateness stochastic scheduling problem to minimize the value-at-risk[J]. Flexible Services and Manufacturing Journal, 2019(31): 472-496.

[15] QIU D H, LI H, SUN J L. Measuring software similarity based on structure and property of class diagram[C]//Proceedings of Sixth International Conference on Advanced Computational Intelligence. [S.l.:s.n.], 2013: 75-80.

[16] WANG Z P, HU H C, CHENG G Z. A DNS framework design based on mimic security defense[J]. Electronic Journal, 2017, 45(11): 6-9.

[17] 胡腾, 李观文, 周华春. 面向服务的数据中心安全框架[J]. 电信科学, 2018, 34(1): 8-16.
HU T, LI G W, ZHOU H C. Service-oriented security framework for datacenter networks[J]. Telecommunications Science, 2018, 34(1): 8-16.

[作者简介]

高明（1979– ），男，博士，浙江工商大学信息与电子工程学院副教授、网络系主任，主要研究方向为新型网络体系架构及工业互联网。

罗锦（1996– ），男，浙江工商大学信息与电子工程学院硕士生，主要研究方向为新型网络体系架构及工业互联网。

周慧颖（1997– ），女，浙江工商大学信息与电子工程学院硕士生，主要研究方向为新型网络体系架构及工业互联网。

焦海（1995– ），男，浙江工商大学信息与电子工程学院硕士生，主要研究方向为新型网络体系架构及工业互联网。

应丽莉（1997– ），女，浙江工商大学信息与电子工程学院硕士生，主要研究方向为新型网络体系架构及工业互联网。