计算机科学
COMPUTER SCIENCE

基于执行体防御能力的拟态防火墙执行体调度算法

刘文贺，贾洪勇，潘云飞

引用本文

刘文贺，贾洪勇，潘云飞.基于执行体防御能力的拟态防火墙执行体调度算法[J]．计算机科学, 2022, 49(11A): 211200296-6.

LIU Wen-he, JIA Hong-yong, PAN Yun-fei. Mimic Firewall Executor Scheduling Algorithm Based on Executor Defense Ability [J]. Computer Science, 2022, 49(11A): 211200296-6.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.211200296

基于执行体防御能力的拟态防火墙执行体调度算法

刘文贺　贾洪勇　潘云飞
郑州大学网络空间安全学院　郑州 450000

摘　要　拟态防御技术是解决现有网络环境“易攻难守”局面的有效手段,拟态防御技术通过提升系统的动态性、异构性和随机性来构建安全可靠的系统。异构执行体的调度是拟态防御的关键环节。已有的调度算法缺乏态势感知能力,只能按照已有策略对执行体进行调度,存在适用性差的问题。为此,提出了一种基于执行体防御能力的调度算法 DCOE。DCOE 基于经典的流量监测算法,识别出当前流量的威胁类型和威胁程度,并根据各执行体针对当前流量的防御能力动态地调整异构执行体的种类和数量。仿真实验表明,DCOE 算法可以在减少对异构执行体调度次数的基础上降低系统的失效率和逃逸率,即在降低系统开销的前提下提升系统的防御水平,增加敌手的攻击难度。

关键词:拟态防御;异构执行体;调度算法;执行体防御能力;仿真
中图法分类号　TP393.08

Mimic Firewall Executor Scheduling Algorithm Based on Executor Defense Ability

LIU Wen-he,JIA Hong-yong and PAN Yun-fei
School of Cyber Science and Engineering,Zhengzhou University,Zhengzhou 450000,China

Abstract　Mimic defense technology is an effective means to solve the easy to attack but difficult to defend situation in existing network environment. Mimic defense technology builds a safe and reliable system by improving the dynamics,heterogeneity and randomness of the system. The scheduling of heterogeneous executive bodies is the key link of mimic defense. Existing scheduling algorithms lack of situational awareness and can only schedule the executor according to the existing strategy,which has the problem of poor applicability. For this reason,DCOE,a scheduling algorithm based on the comprehensive defense capability of the executive body is proposed. Based on the classic traffic monitoring algorithm,DCOE identifies the threat type and threat level of the current traffic,and dynamically adjusts the types and numbers of heterogeneous executives according to the defense capabilities of each executive against the current traffic. Simulation experiments show that,the DCOE algorithm can reduce the failure rate and escape rate of the system on the basis of reducing the number of scheduling heterogeneous executives,that is,improve the overall defense level of the system on the premise of reducing the system overhead,and increase the difficulty of the adversary's attack.

Keywords　Mimic defense,Heterogeneous executor,Scheduling algorithm,Executor defense ability,Simulation

1　引言

随着科技水平的不断进步,互联网在电力、金融、交通、能源等领域扮演的角色也越来越重要。但迅猛发展的背后往往会存在很多安全隐患,网络环境中的未知后门和漏洞往往是这些隐患的最大诱因。传统的网络防御手段通常是单一的、相似的、静态的、无法应对当前“易攻难守”的局势。为此,美国学者提出了移动目标防御[1](Moving Target Defense,MTD)的理念,该方案通过多样化、动态化、随机化的思想来改变现有网络环境静态性、相似性、确定性的特性,以提高攻击方利用漏洞创建和维持攻击链的成本和代价。端信息跳变技术[2]是在端到端的数据传输中通信双方按协定伪随机地改变端口、地址、时隙、加密算法甚至协议等端信息,从而破坏敌方攻击与干扰,实现主动网络防护。网络空间拟态防御[3](Cyber Mimic Defense,CMD)是由邬江兴院士提出的网络防御技术,是改变网络空间“易守难攻”游戏规则的理论,其通过动态非相似余度(Dynamical Heterogeneous Redundant,DHR)机制,来增大系统动态、异构、冗余的属性,以保障系统具有内生安全属性,来增大系统在面对未知漏洞和后门时的可靠性。拟态防御系统架构图如图 1 所示。

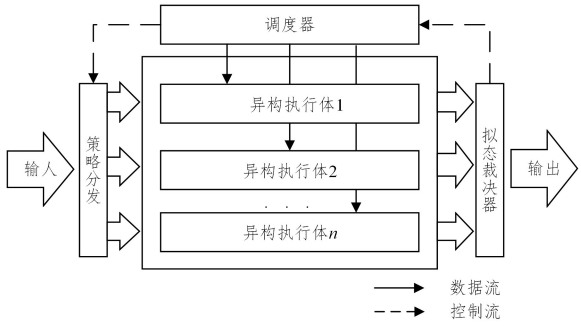

图 1　拟态防御系统通用架构图

Fig. 1　General architecture of mimic defense system

异构执行体的调度是拟态防御的关键环节,是系统多样

基金项目:河南省科技攻关计划(192102210115);郑州市协同创新重大专项(20XTZX-X010)
This work was supported by the Science and Technology Research Plan of Henan Province(192102210115) and Collaborative Innovation Major Project of Zhengzhou City(20XTZX-X010).
通信作者:刘文贺(liuwh9907@163.com)

性、动态性、可靠性的保障。先入先出调度算法 FIFO(First In First Out)基于队列结构,将最先进入等待池的执行体置换入运行池中。文献[4]采用完全随机的调度策略(Completely Random Algorithm,CRA)置换异构执行体,通过增大系统的随机性来降低系统产生相同漏洞的概率,进而增大系统的安全性。文献[5]提出了一种兼顾动态性和可靠性的调度算法,即随机种子最小相似度(Random Seed Minimum Similarity,RSMS),该算法首先随机确定种子冗余体,然后根据相似度指标选择整体相似度最小的异构执行体。该算法在兼顾动态性的同时考虑了异构执行体之间的异构性,提高了系统的安全性和可靠性。文献[6]提出了基于优先级和时间片的执行池调度算法(Pool Scheduling Based on Priority and Time slice,PSPT),该算法对执行池相似性指标和时间片等策略进行调度,综合系统的时间和空间特性,进一步平衡系统的动态性和异构型。文献[7]提出了一种基于最大异构性和安全度的随机种子调度算法(Random Seed & Maximum Heterogeneous and Safety,RSMHS),该算法首先随机选择执行体种子,随后利用深度学习模型选择综合考量最大的调度方案。

现有的执行体调度算法往往从动态、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[8],降低系统存在共同漏洞和后门的概率,进而增加入侵者的攻击难度和成本,但随着异构执行体数量的增加,系统也会愈加复杂,成本也会随之加大,为此需要在安全性和复杂性之间折衷[9]。

当前调度算法缺乏对当前网络状况的感知能力,无法根据网络状况动态调整异构执行体数量。本文在现有架构的基础上提出了一种可以根据场景变化动态地调整异构执行体的种类和数量的调度算法,即基于执行体防御能力的拟态防火墙执行体调度算法(Scheduling Algorithm Based on the Comprehensive Defense Capability of Executives,DCOE),该算法定义综合防御能力值并将各个异构执行体对当前流量的防御能力量化,使用经典的卷积神经网络(Convolutional Neural Networks,CNN)[10]对网络中的实时流量进行检测,得到当前流量的威胁种类和威胁程度,根据当前流量的威胁程度动态调整调度执行体的数量,并根据执行体防御值选取针对当前流量防御能力最强的执行体集合,实现最大的安全增益,最后通过仿真实验验证结果比较算法的综合性能。此外,设置仿真实验验证系统安全性与执行体可靠性之间的关系。

2 防火墙执行体调度指标

异构执行体的调度是拟态防御中的关键环节,现有调度算法往往将异构执行体的安全性和执行体池的异构度视为执行体调度的指标,但这些指标无法感知当前网络环境进而动态地调度执行体,会造成资源的损耗。为此,本文提出了一种衡量执行体对当前流量的防御能力指标,即执行体防御能力,系统根据上述指标生成调度向量供调度器使用。

2.1 拟态防火墙

拟态防火墙是网络空间拟态防御技术的一种应用,通过动态地调度异构执行体可以增强防火墙的动态性和不确定性,变被动防御成为主动防御,增加入侵者的入侵成本,提升了系统面对未知风险和后门的能力。

拟态防火墙主要由请求分发模块、异构执行体、中心调度器、响应多余度表决器以及流量检测模块组成。请求分发模块是网络请求的真实入口,将用户的请求复制后按照调度策略动态转发至异构执行体模块,是实现异构执行体动态变化的前提条件。响应多余度表决器按照既定算法对用户同一请求的多个响应进行表决,得到统一的输出结果,并将上述结果反馈给中心调度器。中心调度器按照调度算法以及反馈信息生成并向请求分发模块发送调度策略。异构执行体池采用不同的构建实现相同的功能,极大地增加了系统的动态性,降低了系统产生共模漏洞的概率。流量检测模块对系统当前流量进行检测,并根据检测结果生成调度参数。其构造如图 2 所示。

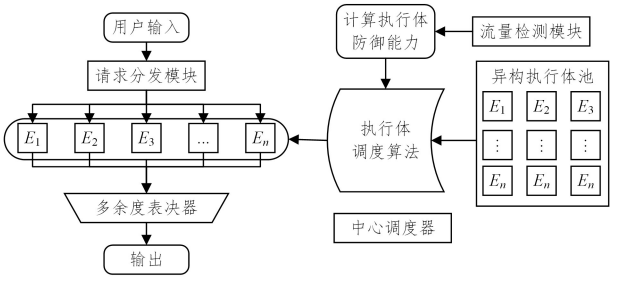

图 2 拟态防御理论构造防火墙架构示意

Fig. 2 Firewall architecture based on mimic defense theory

2.2 执行体防御能力

根据上述描述,构造执行体防御值的相关定义。

定义 1(拟态防火墙执行体) 在拟态构造防火墙系统中,为防火墙系统实现功能的等价构件,记为 $E_i(0\leqslant i\leqslant n)$。

定义 2(调度集) 拟态防火墙中所有的执行体集合,记为 $E=\{E_i \mid i=1,2,3,\cdots,n\}$,其中,$i$ 表示编号,n 表示异构执行体的数量。

定义 3(执行体集) 拟态防火墙系统中同时为系统提供服务且一同参与裁决的异构执行体集合,记为 $R=\{E_i \mid i=1,2,\cdots,r\}$,其中 $|R|=r$ 执行体的余度为 r。

定义 4(威胁值) 入侵检测模型对系统当前流量进行检测的结果,记为 $H=\{h_0,h_1,\cdots,h_j\}$,其中 h_j 为神经网络模型的分类器 Sigmoid 函数的输出结果,$0\leqslant h_0\leqslant 1$ 表示当前流量被流量检测系统判定为正常流量的概率,$0\leqslant h_j\leqslant 1$ 为当前流量被流量检测系统判定为第 j 种攻击模式的概率。

假定某拟态防火墙中入侵检测模型将攻击模式分为 Denial-Of-Service(DOS);Remote to Local(R2L);User to Root(U2R);Surveillance or probe(Probe):检测模型对某条流量检测结果为 $H=\{0.729,0.1010,0.0019,0.1260,0.0280\}$即模型判定该流量类型的概率如表 1 所列。

表 1 入侵检测结果

Table 1 Results of intrusion detection

Type of attack	Probability
Normal	0.7290
DOS	0.1010
R2L	0.0019
U2R	0.1260
Rrobe	0.0280

定义 5(执行体漏洞矩阵) 对于 r 个异构执行体 E_1,E_2,E_3,$\cdots$,E_r 以及包含 ω 个漏洞的集合 V,其中 $M_{E\cdot V}=(msv_{i,j})$ 为对应的执行体-漏洞矩阵,其中:

$$msv_{i,j}=P_{vul} \tag{1}$$

其中,P_{vul} 为异构执行体 i 对攻击方式 j 存在漏洞的概率,P_{vul}

可以通过漏洞评分系统(Common Vulnerability Scoring System，CVSS)得到。假定 $r=7$，$\omega=4$ 时的执行体漏洞矩阵如图 3 所示。

$$\begin{bmatrix} 0.0497 & 0.1281 & 0.0945 & 0.0511 \\ 0.1483 & 0.0772 & 0.1134 & 0.1882 \\ 0.0096 & 0.0877 & 0.1921 & 0.1918 \\ 0.1726 & 0.0552 & 0.0173 & 0.1126 \\ 0.1684 & 0.1120 & 0.1558 & 0.1489 \\ 0.1635 & 0.0010 & 0.1056 & 0.0630 \\ 0.0953 & 0.1332 & 0.1720 & 0.1738 \end{bmatrix}$$

图 3　执行体-漏洞矩阵

Fig. 3　Executive-vulnerability matrix

定义 6(执行体综合防御能力)　根据流量检测结果 H 以及执行体-漏洞矩阵生成防御能力数组，描述执行体对当前流量的防御能力 $DEF=\{def_1,def_2,def_3,\cdots,def_n\}$，其中：

$$def_i=\sum_{kt=1}^{t=j}h_t\times msv_{i,t} \tag{2}$$

在实际应用中，流量检测模块根据异常检测算法检测当前流量的威胁种类和程度得到 H，并根据预先统计的执行体-漏洞矩阵计算出执行体针对当前流量的综合防御能力 DEF。

定义 7(余度)　调度器调度的执行体个数，记作 r，在拟态防火墙系统中可根据入侵检测的结果进行动态调整。

定义 8(调度方案)　系统按照防御能力 def_i 值对异构执行体进行降序排列，并得到调度方案 $F=\{E_k\mid k=1,2,3,\cdots,r\}$。其中，$r$ 为调度方案的余度，序号 k 越小表示该执行体对当前流量的防御能力越强，该执行体应当优先调用。

定义 9(调度方案)　调度器基于上述调度方案 F 动态调整执行体种类和数量，并生成调度向量 $\boldsymbol{D}=\{d_1,d_2,\cdots,d_r\}$，其中：

$$d_i=\begin{cases}0, & \text{执行体被调度}\\ 1, & \text{执行体不被调度}\end{cases} \tag{3}$$

3　基于执行体防御能力的拟态防火墙执行体调度算法

常规的调度方式只能按照预定的程序对执行体进行调度，不能感知当前的网络环境的变换进而动态地调度异构执行体。为了解决上述调度算法存在的问题，将入侵检测与异构执行体的调度结合，首先选择合适的入侵检测数据集供拟态防火墙系统的入侵检测模型进行训练，得到入侵检测模型，随后使用训练的模型判断当前流量的威胁种类和威胁程度，结合执行体-漏洞矩阵生成防御能力数组，拟态调度器根据上述结果动态地调节异构执行体种类和数量，进而在提高系统稳定性的基础上降低由于冗余系统导致的资源损耗，进而寻求最大的系统安全增益。算法流程如图 4 所示。

DCOE 算法的步骤如下：

STEP 0　导入训练后的模型：

将训练后的模型导入到系统中。

STEP 1　生成执行体-漏洞矩阵：

系统根据执行体应对相应攻击方式的能力生成执行体-漏洞矩阵 $M_{S\text{-}V}=(msv_{i,j})_{r\times\omega}$，在实际应用中该矩阵可以通过 CVSS[11] 漏洞库或测试得到。

STEP 2　入侵检测：

使用流量检测模块对当前数据流进行检测，记为 $H=\{h_0,h_1,h_2,\cdots,h_j\}(j\leqslant N)$。

STEP 3　计算余度：

根据流量检测结果 H 得出当前流量的威胁值，判定与阈值 H_{thr} 的关系，并调整余度。

STEP 4　生成防御能力值：

系统根据流量检测结果 H 和执行体-漏洞矩阵生成各个异构执行体的防御能力值 def。

STEP 5　生成调度方案：

系统结合流量检测结果和各个异构执行体的防御能力值判定调度异构执行体的种类和数量，生成调度方案 F。

STEP 6　生成调度向量：根据 STEP5 中的调度方案构造调度向量，调度器根据调度向量对异构执行体进行调度。

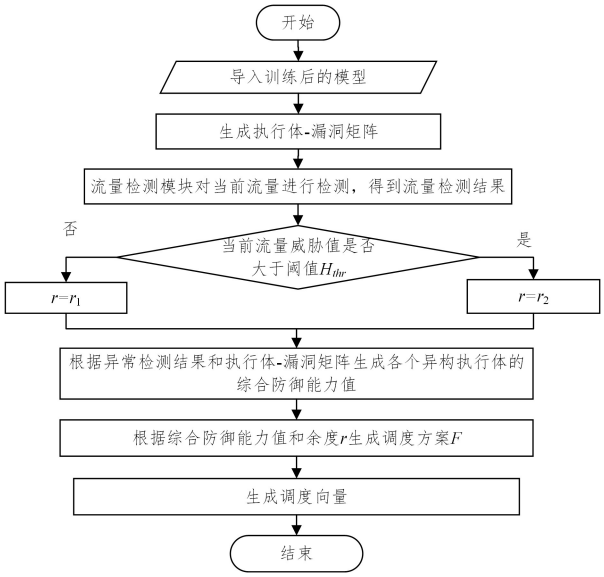

图 4　DCOE 算法流程图

Fig. 4　DCOE algorithm flowchart

算法 1　基于执行体防御能力的调度算法

输入：当前流量 FLOW；异构执行体集 R；执行体-漏洞矩阵 $M_{E\text{-}V}$

输出：调度方案 F

```
1. / * 调度算法方案 * /
2. 初始化：Set F=Φ，Def=Φ
3. / * 初始化调度方案，执行体防御能力数组 H=AD(FLOW) * /
4. / * 流量检测模块对当前流量进行检测 * /
5. if H[0]>H_thr
6. / * 威胁程度大于阈值，余度设为 r_1 * /
7. r=r_1
8. else：
9. / * 否则余度设为 r2 * /
10. r=r_2
11. for i=1；n=size(R)；i≤n；i++
12. / * 初始化执行体防御能力数组 * /
13. def_i=0
14. for k=1；k≤j；k++
15. / * m 为当前分类器对异常流量分类的种类数 * /
16.    def_i=def_i+(msv_{i,k}×h_k)
17. end for
18. end for
19. Ascending Sort(Dispatch F)
20. / * 对执行池按 def 值进行升序排序 * /
21. return Dispatch F[：r]
22. / * 输出调度方案 * /
```

4 仿真实验

为了比较传统的调度算法 FIFO,CRA,RSMS 以及本文提出的 DCOE 算法的算法性能,设计了仿真实验,分别对不同调度算法的效率、准确性、可靠性进行比较。实验平台主机配置为 Intel(R) Core(TM) i5-9500 CPU@ 3.00 GHz 3.00 GHz, RAM 16.0 GB,编程语言为 Python。

4.1 仿真建立

在仿真实验中,为了模拟真实的网络环境的攻击状况,使用的数据集应当贴近真实的网络环境,在选取数据集时应当考虑选取具有构建网络的完整性、通信数据和攻击的多样性、多个设备的异构性和真实性特性的数据集[12]。

真实世界中的数据往往都是无法直接供深度学习模型训练的数据,为此需要在训练之前对数据进行预处理操作。典型的数据预处理方式有:数据清理、数据集成、数据变化、数据归纳等[13]。通过数据预处理可以大大降低训练成本并提高模型的准确率和可靠性。

KDD99 数据集的每条数据包括 41 个特征位和 1 个标志位。特征位包括 38 个数字特征和 3 个符号特征,标志位表示该数据是否为异常数据。

为了将数据集转换为可以供模型训练的数据,需要对数据集进行预处理操作。本文使用 Python 软件进行数据预处理工作,对数据集进行预处理后,成为 494 021×122 和 311 029×122 维的数据即为输入数据,如表 2 所列。

表 2 KDD 数据集类型及数量

Table 2 Type and number of KDD data set

Type of attack	10%KDD	Corrected KDD
All Type	494 021	311 092
Normal	97 278	60 593
DOS	391 485	229 853
R2L	4 107	4 166
U2R	52	228
Rrobe	1 126	16 189

为了衡量不同调度算法之间的差异性,分别设定如下衡量调度算法的指标。

通过比较不同调度算法对异构执行体的调度次数可以衡量相应调度算法的效率;通过比较不同调度算法的失效率以及逃逸率可以比较算法的可靠性和稳定性。

具体仿真步骤如下:

Step 1 初始化执行体集和调度集。

Step 2 使用上述数据集构造威胁输入。

Step 3 根据调度算法从执行体集中选取异构体填入调度集并对当前调度异构执行的数量计数。

Step 4 裁决器对当前执行体的执行结果进行裁决,并输出裁决结果。

Step 5 计算当前调度算法对异构执行体的调度次数并求和。

Step 6 计算调度算法的各项效率。

在实际应用中,执行体-漏洞矩阵可以依据 CVE 漏洞库构建。本文的仿真实验为了进一步验证算法的可靠性与稳定设定执行体产生漏洞的概率符合 β 分布,不同参数的 β 分布的概率密度曲线如图 5 所示。在实验中分别设定不同分布的概率密度参数,用于模拟不同环境下的执行体的安全性。首先选取参数为(1,10)的 β 分布,将其作为常规状态下系统产生漏洞的概率,对 FIFO,CRA,RSMS,DCOE 算法的执行体调度次数、系统失效率、系统逃逸率进行记录并分析 4 种算法的性能。

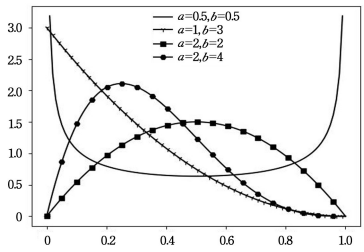

图 5 β 分布概率密度曲线

Fig. 5 Curve of β distribution probability density

4.2 调度次数对比

在拟态防御系统架构中,随着异构执行体数量的增加,系统的安全性也会随之增加,这也会增加系统的复杂性。为此可以通过比较执行体的调度次数来比较调度算法的效率。在实验中,对调度器调度异构执行体的次数进行统计,实验结果如图 6 所示。

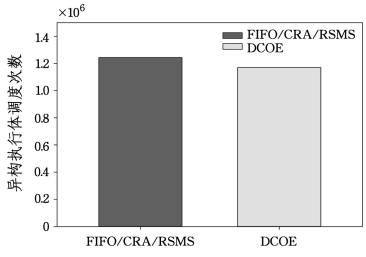

图 6 不同算法的调度次数

Fig. 6 Scheduling times of different algorithms

FIFO,CRA,RSMS 算法在调度的过程中执行体的余度 r 是恒定不变的,具体参数如表 3 所列。在整个仿真过程中异构执行体调度的总次数为流量条数 k 和余度 r 的乘积。本轮实验中,调用异构执行体的总次数为 $I=k\times r$,即 1 244 108 次。

表 3 执行体余度表

Table 3 Redundancy of executor

Type of algorithm	Degree of threat	Redundancy
FIFO&CRA&RSMS	$t\leqslant 0.10$	4
	$t>0.10$	4
DCOE	$t\leqslant 0.10$	3
	$t>0.10$	4

DCOE 算法,需要根据当前流量的威胁程度动态地对执行体的余度 r 进行调整,设定动态调整参数如表 3 所列。本轮实验中,流量总数 $k=311\,027$,余度 r 根据当前流量的威胁程度依据表动态调整,调用异构执行体的总次数为 $I=k\times r$,即 1 170 308 次。

实验结果表明,基于执行体防御能力的拟态防火墙执行体调度算法在仿真实验中对异构执行体的调度次数较其他算法有明显降低,这表明本文算法可以根据当前流量威胁程度动态调整异构执行体的数量,进而提高系统效率。

4.3 失效率对比

在拟态防御理论中,系统的失效率指判决器输出错误结果的集合,即错误结果对应集合的置信度最大。为此,在仿真实验中可以通过裁决器输出错误结果的概率来计算系统的失效率。

在真实的网络环境中,假定异构执行体的异常输出都是由入侵者攻击导致的,“攻击者攻击成功”等价于“判决器判决

法则失效”，“判决器判决法则失效”等价于输出一致错误的执行体个数大于输出一致正确的执行体个数”[14]，即：

$$P_{\text{attacksuccess}}=P(N_{\text{error}}\geqslant N_{\text{correct}})$$

其中，N_{error}指输出一致错误的执行体个数，N_{correct}指输出一致正确的执行体个数。但由于执行体之间由于系统、环境、开发语言等存在的异构性，即使该执行体存在漏洞，但攻击者无法使执行体产生一致性错误输出，则该攻击模式仍然不成功，引入文献[14]的计算公式：

$$\begin{aligned}p_{\text{attacksuccess}}&=P(\text{attacksuccess})\leqslant P(\text{intrusionsuccess})\\&=p_{\text{intrusionsuccess}}\end{aligned}\tag{6}$$

本文中约定 $P_{\text{attack success}}=P_{\text{intrusion success}}$，即系统存在漏洞的概率与系统被攻击成功的概率相同。根据上述描述，在没有先验知识的条件下，根据漏洞产生概率 p_{vul} 生成漏洞。p_{vul} 符合参数为(1,10)的 β 分布，不同调度算法的失效率结果如表 4 所列。

表 4　不同调度算法的失效率

Table 4　Failure rate of different scheduling algorithms

algorithm	Failure rate
FIFO	0.0746
CRA	0.0535
RSMS	0.0144
DCOE	0.0102

为上述实验结果绘制柱状图，如图 7 所示。

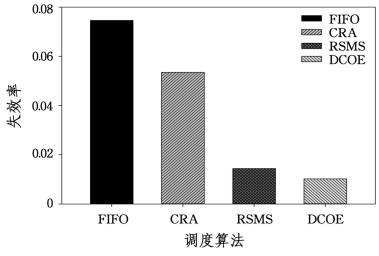

图 7　不同调度算法的失效率

Fig. 7　Failure rate of different scheduling algorithms

4.4　逃逸率对比

由于网络空间具有“设计缺陷不可避免”“后门无法杜绝”“漏洞后门尚不可彻查的属性”[15]，不同的执行体间可能存在较高的相似性，系统之间可能由于设计缺陷存在共生漏洞，这会导致共模逃逸的产生，引发系统故障。因此，为了实现系统的稳态可用，降低攻击逃逸概率，需要采用合适的调度策略算法。

为了比较不同算法之间逃逸率的差异，设计验证系统产生共模逃逸的概率，通过比较系统逃逸率的方式可以验证系统的可靠性。

设执行体 E_n 被攻击成功的概率为 P_n，采用文献的拟态逃逸概率计算方法，拟态逃逸概率计算式如下：

$$P=\prod_{i=1}^{n}p_i\tag{7}$$

其中，P_1 到 P_n 分别表示第 1 号执行体到第 n 号执行体被攻击成功的概率。仿真结果如表 5 所列。

表 5　不同算法的逃逸率

Table 5　Escape rate of different scheduling algorithms

Type of algorithm	Escape rate
FIFO	6.900×10^{-6}
CRA	3.870×10^{-6}
RSMS	0.446×10^{-6}
DCOE	0.581×10^{-6}

为上述实验结果绘制柱状图，如图 8 所示。

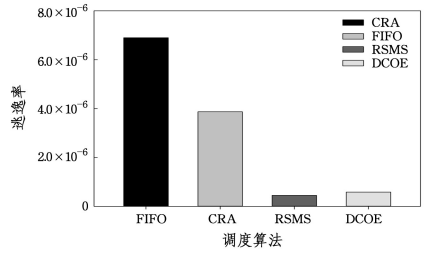

图 8　不同调度算法的逃逸率

Fig. 8　Escape rate of different scheduling algorithms

根据上述实验得出结论：RSMS 算法由于兼顾了动态性和可靠性，具有最低的逃逸率，DCOE 算法在降低异构执行体调度次数的基础上系统的逃逸率较 CRA 和 FIFO 算法有明显的降低，即降低了系统产生共模逃逸的概率，在可靠性和效率之间达到了平衡。

通过仿真实验对上述算法的失效率、逃逸率以及执行体调度次数进行比较，并对算法的不同性能进行排序，结果如表 6 所列。

表 6　4 种算法的性能排序

Table 6　Performance ranking of 4 algorithms

Type of algorithm	Reliability	Efficiency
CRA	较低	中
FIFO	低	中
RSMS	高	低
DCOE	高	高

综合上述实验可以得出结论，DCOE 算法在减少异构执行体调度次数的基础上可以降低系统的失效率、逃逸率，即在降低系统功耗的前提上提高系统的可靠性和稳定性。

4.5　算法在不同漏洞概率下的性能对比

为了进一步验证上述调度算法在不同场景下的性能和表现状况，本节实验调整执行体产生漏洞的概率 P_{vul}，用于模拟具有不同失效率的执行体，分别设定概率密度参数为(1,10)，(1,7)，(1,5)，(1,3)，(0.5,0.5)，(3,1)，记录系统的系统失效率，如图 9 所示。

由图 9 可以看出，DCOE 的系统失效率最低，其次是 RSMS，FIFO 和 CRA 各有优劣，这表明本文算法在执行体安全度不同的状况下具有最低的失效率，即最高的可靠性。上述 4 种算法的系统逃逸率状况如图 10 所示。

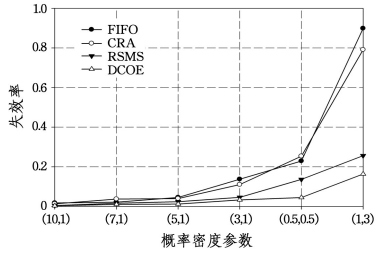

图 9　不同调度算法的失效率

Fig. 9　Failure rate of different scheduling algorithms

由图 9 可以得知，RSMS 算法具有最低的逃逸率，DCOE 算法的逃逸率略高于 RSMS 算法，上述两种算法的逃逸率远低于其他调度算法，但由于 DCOE 算法对执行体调度执行个数小于 RSMS 算法且 RSMS 在进行调度时需要计算执行体的异构度，算法复杂性较高，因此 DCOE 的算法

具有最高的综合性能。

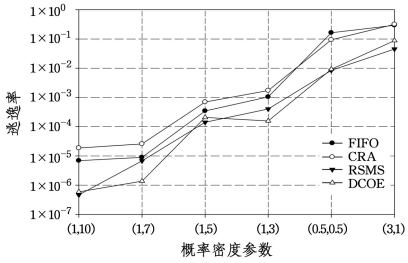

图 10 不同调度算法的逃逸率

Fig. 10 Escape rate of different scheduling algorithms

综合上述实验可以得出结论，本文提出的基于执行体防御能力的拟态防火墙执行体调度算法，在减少异构执行体调度次数的基础上可以降低系统的失效率、逃逸率，即在降低系统功耗的前提下提高了系统的可靠性和稳定性。此外，拟态防御技术虽然大大提升了系统的安全性，但异构执行体的安全程度依然会对系统的安全性造成影响，为此，在调整执行体调度算法的同时也应开展研究，以提高执行体的安全性。

结束语 拟态防御是改变现有网络安全格局的新思想，而异构功能等价体的调度是其中的关键环节。针对现有调度算法缺乏态势感知能力、存在适用性差的问题，本文提出了一种基于执行体防御能力的拟态防火墙执行体调度算法，并通过仿真实验验证了该算法的效率。实验结果表明：DCOE 算法能在提升系统效率的基础上降低系统的失效率、逃逸率，从而降低共模逃逸的风险。然而，入侵检测模型的选取会对系统的效能产生一定的影响，在今后的研究中可以通过优化入侵检测算法进一步优化系统的性能。此外，由实验可知，执行体的安全性会对拟态系统的整体可靠性产生影响，为此，在工程实践中需要提高执行体的安全性，以提高系统的可靠度。

参 考 文 献

[1] National Science and Technology Council. "Trustworthy cyberspace: Strategic plan for the federal cybersecurity research and development program" [OL]. https://www.nitrd.gov/SUBCOMMITTEE/csia/ed_Cybersecurity_RD_Strategic_Plan_2011.pdf.

[2] SHI L Y, JIA C F, LYU S W. Research onendhopping for active network confrontation [J]. Journal on Communications, 2008 (2): 106-110.

[3] WU J X. Research on Cyber Mimic Defense[J]. Journal of Cyber Security, 2016, 1(4): 1-10.

[4] WU JX, LI J F, ZHANG F, et al. A heterogeneous redundancies scheduling equipment and method[P]. China, CN106161417A, 2016-11-23.

[5] 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.

[6] PU LM, 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.

[7] GAO Y, ZI C C, FENG S F, et al. Security Scheduling Algorithm for Web Gateways Based on Mimicry Defense Theory[J]. Journal of Chinese Computer Systems, 2021, 42(9): 1913-1919.

[8] ZHU Z B, LIU Q R, LIU D P, et al. Research progress of mimic multi-execution scheduling algorithm[J]. Journal on Communications, 2021, 42(5): 179-190.

[9] WEI S, YU H, GU Z Y, et al. Architecture of mimic security processor for industry control syste[J]. Journal of Cyber Security, 2017, 2(1): 54-73.

[10] LECUN Y, BOTTOU L, BENGIO Y, et al. Gradient based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[11] FIRST. Common vulnerability scoring system(Version-3.1) [EB/OL]. (2019-01-01)[2021-03-16]. https://www.first.org/cvss/calculator/3.1.

[12] GHARIB A, SHARAFALDIN I, LASHKARI A H, et al. An Evaluation Framework for Intrusion Detection Dataset [C]// 2016 International Conference on Information Science and Security(ICISS). IEEE, 2017.

[13] LIU M J, WANG X F, HUANG Y L. Preprocessing in data mining[J]. Computer Science, 2000, 27(4): 56-59.

[14] HU H C, CHEN F C, WANG Z P. Performance evaluationson DHR for cyberspace mimic defense[J]. Journal of Cyber Security, 2016, 1(4): 40-51.

[15] WU J X. Robust control and endogenous safety[J]. CivIntegration on Cyberspace, 2018(3): 23-27.

LIU Wen-he, born in 1999, postgraduate. His main research interests include cyber security and mimic defense.