基于高阶异构度的执行体动态调度算法

贾洪勇[1]，潘云飞[1]，刘文贺[1]，曾俊杰[1]，张建辉[2]
（1. 郑州大学网络空间安全学院，河南 郑州 450001；2. 国家数字交换系统工程技术研究中心，河南 郑州 450001）

摘 要：针对当前动态异构冗余系统中异构体调度缺乏动态性和仅考虑二阶异构性，导致系统易被攻击者找到共模漏洞从而攻破系统的问题，提出了一种同时考虑执行体高阶异构度和历史信息的异构执行体动态调度算法——基于高阶异构度的负反馈调度算法。该算法首先计算等待池中执行体的高阶异构度矩阵，然后在每次调度执行体时同时考虑历史威胁和异构体间的高阶异构度来确定调度执行体集。实验表明，结合高阶异构度和历史信息的策略使算法获得了动态性和安全性的平衡，且防御能力较先前算法更优秀。

关键词：拟态防御；高阶异构度；负反馈；动态异构冗余；执行体调度
中图分类号：TP393.08
文献标志码：A
DOI: 10.11959/j.issn.1000−436x.2022055

Executive dynamic scheduling algorithm based on high-order heterogeneity

JIA Hongyong[1], PAN Yunfei[1], LIU Wenhe[1], ZENG Junjie[1], ZHANG Jianhui[2]
1. School of Cyber Science and Engineering, Zhengzhou University, Zhengzhou 450001, China
2. National Digital Switching System Engineering & Technological R&D Center, Zhengzhou 450001, China

Abstract: At present, most isomer scheduling models in DHR systems only considered second-order isomerism and lack dynamics. In order to solve these problems, a dynamic scheduling algorithm was proposed, which combined heterogeneity and dynamics by determining a scheduling scheme with maximum scheduling value. The value was obtained by combining high-order heterogeneity and historical information. The experimental results show that the proposed algorithm has better defense capabilities than other scheduling algorithms, and its strategy can achieve a balance of dynamics and security.

Keywords: mimic defense, high-order heterogeneity, negative feedback, dynamic heterogeneous redundancy, executive scheduling

0 引言

针对当今网络安全易攻难守的局面，美国国家科学技术委员会提出了移动目标防御（MTD, moving target defense）[1]。其采用包括 IP 地址和端口跳变[2]、IP 安全协议随机化[3]、用户角色随机化、地址空间随机化[4-7]、指令集合随机化[8]以及动态路由[9]等相关技术手段构建一种动态的、不确定的网络攻防环境，使防御目标对攻击者表现出不可预测的状态。

2014 年，邬江兴院士[10]提出了网络空间拟态防御（CMD, cyber mimic defense），并结合生物学“拟态伪装”和“特异和非特异性免疫”现象，以异构冗余和动态反馈机制不断调整防御系统执行环境，

收稿日期：2021−11−01；**修回日期**：2022−01−18
通信作者：潘云飞，PanYF0511@163.com
基金项目：河南省科技攻关计划基金资助项目（No.192102210115）；郑州市协同创新重大专项基金资助项目（No.20XTZX-X010）
Foundation Items: Science and Technology Research Plan of Henan Province (No.192102210115), Collaborative Innovation Major Project of Zhengzhou (No.20XTZX-X010)

从而打破以往的被动防御，以一种主动变迁[11]的方式对已知或未知漏洞后门等实现优秀的防御效果。其核心架构是动态异构冗余（DHR, dynamic heterogeneous redundancy）[12-13]，如图 1 所示。

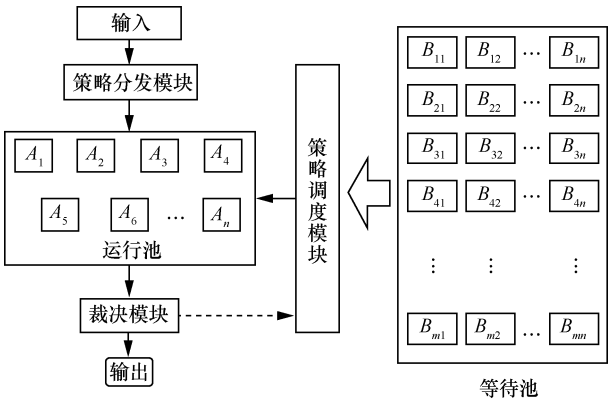


图 1 DHR 架构

调度和裁决模块是 DHR 的 2 个核心模块。调度模块具有从等待池调度执行体进入运行池的功能，一个优秀的调度算法应兼顾调度的动态性和安全性，使攻击者无法判断当前运行执行体编号并使调入执行体对攻击者的攻击造成严重阻碍；裁决模块具有判断运行池中执行体输出是否正确的功能，其能帮助系统分辨出已被攻破的执行体，评估并预警系统的安全隐患。调度/裁决算法的可靠程度关系到 DHR 架构的安全性，即整个系统的安全程度，因此对调度/裁决算法的研究是至关重要的。本文主要针对调度算法的优化进行研究。

目前，围绕调度算法[14]的优化问题，已有很多学者提出了相应的解决方案。文献[15]提出了基于历史信息的人工调度算法 FAWA（feedback artificial weighted algorithm），该算法通过历史记录威胁信息动态调度执行体以达到动态改变的效果，但未考虑执行体间的差异性问题；文献[16-17]采用二阶相似性进行裁决/调度算法优化，这种方法能有效增大运行池中执行体的异构度，减小共模漏洞，但在等待池中执行体确定的情况下会造成执行体集调度单一化的问题；文献[18]提出了基于随机种子的调度算法，该算法考虑了调度的动态性和执行体间异构度，但其只考虑了执行体的二阶异构度，无法准确评估执行体间的差异性；文献[19]提出了高阶异构度的性质并给出了基于高阶异构度的裁决算法，其证明了引入高阶异构度的裁决算法具有优于传统仅考虑二阶异构度算法的安全性，但并未给出高阶异构度的计算方法。目前，执行体调度领域的研究缺少一种从高阶层面考虑执行体间异构度的动态调度算法。

本文的主要研究工作及贡献如下。

1) 在 FAWA 的基础上引入高阶异构度思想，设计并提出了一种同时考虑历史威胁信息和执行体高阶异构度的调度算法 HFAWA（high level heterogeneity feedback artificial weighted algorithm），解决了因未考虑异构度导致系统共模漏洞过多、只考虑异构度造成执行体调度固化无法动态变动的问题。

2) 基于容斥原理给出高阶相似度和高阶异构度的计算方式，解决了目前缺少高阶异构度计算方法的问题。

3) 通过对 DHR 结构中调度模块与裁决模块间关联分析，提出了根据裁决算法动态改变的 HFAWA 改进策略，并以大数裁决算法为例给出了相应改变方法。

4) 通过进行碰撞实验和大数裁决系统攻击（LSA, large number adjudication system attack）实验，对现有 5 种调度方案进行对比评估，验证了 HFAWA 相较于以往方案具有明显的安全性优势且兼具动态调度执行体的能力。

1 基于高阶异构度的负反馈调度算法

1.1 二阶异构度存在的问题

目前，在引入异构度的执行体调度算法的研究中，使用的异构度指标均为二阶异构度[16]，其通过异构体之间两两比较的差异程度值求和的方式来计算整体的差异度。当运行池大小大于 2 时，仅考虑二阶异构度的调度算法会产生局限性，造成系统出现共模漏洞。下面进行举例分析。

假设在系统等待池中存在 4 个执行体（编号为 1～4），其存在的漏洞情况如表 1 所示。

表 1 执行体存在的漏洞情况

执行体编号	漏洞编号
1	1,2,3,4,5
2	3,5,7,8,9
3	1,5,9,10,11
4	1,2,7,8,9

若当前所需调度执行体数量为 3，则可以得到异构体间的 2 阶相似度矩阵为

$$\boldsymbol{S}_{1,2,3,4}^{(2)}=\begin{bmatrix}5&2&2&2\\2&5&2&3\\2&2&5&2\\2&3&2&5\end{bmatrix}$$

对各种调度情况的相似度进行求和，如表2所示。

表 2　　二阶相似度求和

调度编号	相似度值
1,2,3	6
1,2,4	7
2,3,4	7
1,3,4	6

算法应选取二阶相似度最小（即异构度最大）结果作为调度方案，即 1,2,3 或 1,3,4。但当观察执行体漏洞情况时发现，无论是 1,2,3 还是 1,3,4 均存在大小为 1 的共模漏洞（5 或 1）；当系统选择 1,2,4 时，系统整体共模漏洞为 0，为最优解。因此，需要引入高阶异构度来解决这一问题。

1.2　基于相似度的高阶异构度计算方法

文献[18]给出了二阶相似度的相关含义指标，即在余度为 n 的 n 模冗余体集合 Ω 中，相似度由集合中所有不同元素两两之间的相似度之和归一化表示，即

$$S\big|_{\Omega}=\frac{1}{\mathrm{C}_n^2}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}h_{ij} \tag{1}$$

这种相似度计算方式缺乏对高阶异构性的考虑，没有分析高阶共生漏洞对异构体的影响，因此文献[19]提出了高阶相似度的相关含义，并给出了相关性质和推论。本节将在其基础上通过对容斥原理加以分析，从而提出高阶相似性的计算方式，并延伸至高阶异构度的计算。

容斥原理是统计学中一种常用的计数方法，在计数时为了防止重叠部分被重复计算，提出先对整体所有元素进行计数，然后排除被重复计算元素的方法。其数学表达式为

$$|A_1\cup A_2\cup\cdots\cup A_n|=\sum_{i=1}^{n}|A_i|-\sum_{1\leqslant i<j\leqslant n}|A_i\cap A_j|+\sum_{1\leqslant i<j<k\leqslant n}|A_i\cap A_j\cap A_k|-\cdots+(-1)^{n-1}|A_1\cap A_2\cap\cdots\cap A_n| \tag{2}$$

$$|\overline{A_1}\cup\overline{A_2}\cup\cdots\cup\overline{A_n}|=|S|-\sum_{i=1}^{n}|A_i|+\sum_{1\leqslant i<j\leqslant n}|A_i\cap A_j|-\sum_{1\leqslant i<j<k\leqslant n}|A_i\cap A_j\cap A_k|+\cdots+(-1)^n|A_1\cap A_2\cap\cdots\cap A_n| \tag{3}$$

在碰撞实验中，执行体由防御原子组成，因此可以将执行体的防御面积当作该执行体中防御原子的集合，从而通过容斥原理计算多个执行体的共同防御面积，即相似性，如图 2 所示。

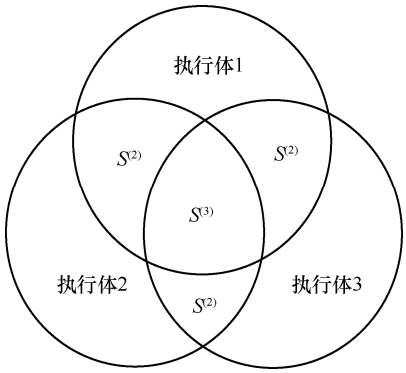


图 2　三模异构冗余体的高阶相似度

图 2 中以圆的面积表示执行体的防御平面，$S^{(2)}$表示 2 个执行体之间的二阶相似区域，$S^{(3)}$表示 3 个执行体之间的三阶相似区域，同理，$S^{(n)}$表示 n 个执行体之间的 n 阶相似区域，它们之间的关系为

$$S^{(n)}\in S^{(n-1)}\in\cdots\in S^{(2)}\in\boldsymbol{S} \tag{4}$$

易知 n 阶相似区域性质与 n 阶相似度性质相符，因此，可将图 2 中的 n 阶相似区域等价为 n 阶相似度进行计算。

定义 1　高阶相似度计算方式。n 模异构冗余体的防御面积为 $P^{(n)}=|A_1\cup A_2\cup\cdots\cup A_n|$，$|A_i|=S^{(1)}(1\leqslant i\leqslant N)$，$|A_i\cap A_j|=S^{(2)}(1\leqslant i<j\leqslant N)$，…，$|A_1\cap A_2\cap\cdots\cap A_n|=S^{(n)}$，则 n 模相似度计算式为

$$S^{(n)}=\left|\sum_{i=1}^{\tau_1}S_i^{(1)}-\sum_{i=1}^{\tau_2}S_i^{(2)}+\cdots+(-1)^{n-1}\cdot\sum_{i=1}^{\tau_{n-1}}S_i^{(n-1)}-\left|A_1\cup A_2\cup\cdots\cup A_n\right|\right|=\left|\sum_{j=1}^{n-1}\sum_{i=1}^{\tau_j}(-1)^{j-1}S_i^{(j)}-P^{(n)}\right| \tag{5}$$

在得到高阶相似度 $S^{(n)}$之后，可进行计算求得高阶异构度[20-22]。以碰撞实验为例，高阶异构度的

计算式为

$$H^{(n)} = P^{(n)} - S^{(n)} \tag{6}$$

对其进行归一化处理，有

$$\overline{H}_i^{(n)} = \frac{H_i^{(n)}}{\sum_{i=1}^{\tau_t} H_i^{(n)}} \tag{7}$$

1.3 算法内容

针对之前所述调度算法存在的问题，基于 FAWA 和高阶异构度思想，设计了一种同时考虑历史威胁信息和异构体间高阶异构度的动态调度算法 HFAWA。符号表示如表 3 所示，并根据以下定义对算法进行形式化描述。

表 3 符号表示

符号	定义
Ω	异构执行体池
N_{W}	等待池执行体容量
N_{R}	运行池执行体容量
A_i	异构执行体 i，$1<i<N_{\mathrm{W}}$
$\boldsymbol{P}$	高阶防御面积矩阵
$\boldsymbol{S}$	高阶相似度矩阵
$\boldsymbol{H}$	高阶异构体矩阵
$\boldsymbol{M}$	执行体调度概率矩阵
D	调度值
Sel	选取执行体集

定义 2 异构执行体池。异构执行体池由多个功能等价的异构执行体组成，异构执行体池分为等待池 $\boldsymbol{\Omega}_{\mathrm{W}}$ 和运行池 $\boldsymbol{\Omega}_{\mathrm{R}}$，可表示为 $\boldsymbol{\Omega}=\{A_1,A_2,\cdots,A_N\}$，各执行体间相互独立。

根据表 3 中的符号定义，对其中部分符号的用法及计算方式做以下说明。

1) $P_{\mathrm{Sel}}^{(n)}$ 为 n 模异构体集 Sel 的防御面积，如 $P_{1,2,4}^{(3)}$ 为 3 模异构体{1,2,4}的防御面积，计算方式为 $A_1 \cup A_2 \cup A_4$。

2) $S_{\mathrm{Sel}}^{(n)}$ 为 n 模异构体集 Sel 的相似度，如 $S_{1,2,4}^{(3)}$ 为 3 模异构体{1,2,4}的相似度，计算方式为容斥原理。

3) $H_{\mathrm{Sel}}^{(n)}$ 为 n 模异构体集 Sel 的异构度，如 $H_{1,2,4}^{(3)}$ 为 3 模异构体{1,2,4}的异构度，在碰撞实验中异构度计算式同式(6)。

4) $\boldsymbol{M}$ 为执行体调度概率矩阵，其值根据历史威胁反馈动态改变，计算方式与文献[14]中计算方法相同。

5) 调度值 D 是算法选取执行体的重要凭证，D 值越大，执行体被调度的可能性越高。D 的计算式为

$$D = M_i + H_{\{\mathrm{Sel}+A_i\}}^{(n)} \tag{8}$$

根据以上定义及计算式，基于高阶异构度的负反馈调度算法 HFAWA 如算法 1 所示。

算法 1 基于高阶异构度的负反馈调度算法 HFAWA

输入 等待池 $\boldsymbol{\Omega}_{\mathrm{W}}$，运行池执行体容量 N_{R}，高阶异构度矩阵 $\boldsymbol{H}$，执行体调度概率矩阵 $\boldsymbol{M}$，选取执行体集 Sel，最大调度值集合 best_s。其中 $H^{(n)}$ 已提前计算完毕，$\boldsymbol{M}$ 矩阵初始默认值为全 1，Sel 与 best_s 初始为空，大小与 $\boldsymbol{\Omega}_{\mathrm{R}}$ 相同

输出 运行池 $\boldsymbol{\Omega}_{\mathrm{R}}$

1) 计算 $\boldsymbol{M}$ 中最大值所对应执行体，存于矩阵 $\boldsymbol{M}_{\max}$
2) for A_i in $\boldsymbol{M}_{\max}$
3) 将 A_i 加入 Sel
4) for n in $\{2,3,\cdots,N_{\mathrm{R}}\}$
5) for A_{W} in $\{\boldsymbol{\Omega}_{\mathrm{W}}-\mathrm{Sel}\}$
6) 计算 $D^{(n)} = \boldsymbol{M}_{\mathrm{W}} + H_{\{\mathrm{Sel}+A_{\mathrm{W}}\}}^{(n)}$
7) if $D^{(n)} > \mathrm{best\_s}_n$
8) $\mathrm{best\_s}_n \leftarrow D^{(n)}$
9) Sel=Sel + A_{W}
10) 若 $n \neq N_{\mathrm{R}}$，将 Sel 送至步骤 4)
11) end if
12) end for
13) end for
14) end for
15) 输出 $\mathrm{best\_s}_{N_{\mathrm{R}}}$ 对应的 Sel 至 $\boldsymbol{\Omega}_{\mathrm{R}}$
16) 输出 $\boldsymbol{\Omega}_{\mathrm{R}}$

算法 1 为一次调度的 HFAWA 实现，其中矩阵 $\boldsymbol{M}$ 随每次调度记录的历史威胁信息反馈而改变，从而影响下次调度结果。算法 1 对应的算法流程如图 3 所示。

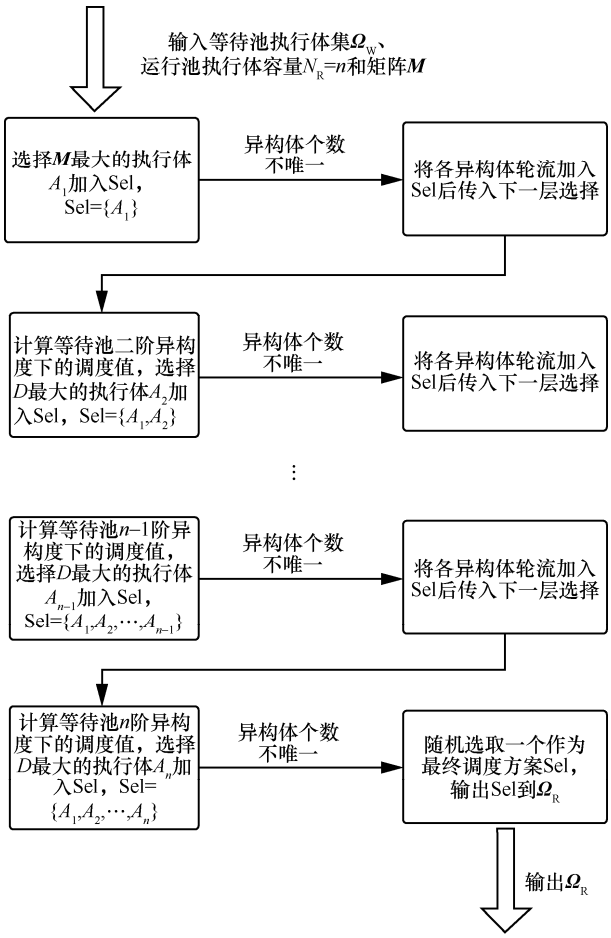


图 3　算法 1 对应的算法流程

1.4　算法用例

下面以 $N_{\mathrm{R}}=5$ 的 DHR 系统为例，对 HFAWA 运行流程进行简要说明。

1) 当选择执行体集 Sel 为空时，挑选 $\boldsymbol{M}$ 数组中最大的执行体 A_1 进行调度加入 Sel。$\mathrm{Sel}=\{A_1\}$，将 Sel 送入算法进行下一执行体选择。

2) 将等待池中除Sel中已有执行体外的执行体 w_i 轮流与 Sel 中执行体匹配，计算 $D=H^{(2)}_{\{A_1,w_i\}}+\boldsymbol{M}[i]$，选择 D 最大的执行体加入 Sel；若 D 最大值不唯一，将对应执行体分别加入 Sel。$\mathrm{Sel}=\{A_1,A_2\}$，将 Sel 送入算法进行下一执行体选择。

3) 将等待池中除Sel中已有执行体外的执行体 w_i 轮流与 Sel 中执行体匹配，计算 $D=H^{(3)}_{\{A_1,A_2,w_i\}}+\boldsymbol{M}[i]$，选择 D 最大的执行体加入 Sel；若 D 最大值不唯一，将对应执行体分别加入 Sel。$\mathrm{Sel}=\{A_1,A_2,A_3\}$，将 Sel 送入算法进行下一执行体选择。

4) 将等待池中除 Sel 中已有执行体外的执行体 w_i 轮流与 Sel 中执行体匹配，计算 $D=H^{(4)}_{\{A_1,A_2,A_3,w_i\}}+\boldsymbol{M}[i]$，选择 D 最大的执行体加入 Sel；若 D 最大值不唯一，将对应执行体分别加入 Sel。$\mathrm{Sel}=\{A_1,A_2,A_3,A_4\}$，将 Sel 送入算法进行下一执行体选择。

5) 将等待池中除 Sel 中已有执行体外的执行体 w_i 轮流与 Sel 中执行体匹配，计算 $D=H^{(5)}_{\{A_1,A_2,A_3,A_4,w_i\}}+\boldsymbol{M}[i]$，选择 D 最大的执行体加入 Sel；若 D 最大值不唯一，从中随机选取一执行体加入 Sel。$\mathrm{Sel}=\{A_1,A_2,A_3,A_4,A_5\}$，将 Sel 输出，Sel 即当次选择调度 5 模异构体。

1.5　算法分析

目前，针对多执行体策略的评估方法有很多，如基于博弈模型对攻防双方进行收益量化作为评判标准[23]、采用攻击成功累计时长作为评判标准[24]、采用成功控守目标个数作为评判标准[25]等。本文采用当前运行池中执行体集的共模漏洞个数作为评判标准。

在不考虑执行体历史信息的影响，即执行体调度概率矩阵 $\boldsymbol{M}$ 时。算法的调度值仅受异构度值 H_i 影响，即

$$D=H^{(n)}_{\{\mathrm{Sel}+A_i\}} \tag{9}$$

以表 1 中执行体情况为例进行分析。

1) 在算法初始选择执行体时，由于各执行体间无调度值差别，因此遍历所有执行体分别加入执行体待选集 Sel 中，并送入算法进行下一阶段执行体选择，$\mathrm{Sel}=\{\{1\},\{2\},\{3\},\{4\}\}$。

2) 根据之前送入的执行体待选集，分别遍历剩余执行体并计算二阶异构度，选出异构度最高的集合送入算法进行下一阶段执行体选择。根据表 1 中执行体的二阶相似度矩阵可知，此时被选择送入算法的执行体集为 $\mathrm{Sel}=\{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{3,4\}\}$（相似度最低）。

3) 根据之前送入的执行体待选集，分别遍历剩余执行体并计算三阶异构度，选出异构度最高的集合，此时算法计算最高阶数与运行池容量相等，因此输出算法最终选择结果 $\mathrm{Sel}=\{1,2,4\}$，共模漏洞数量为 0。

算法中影响调度值的另一部分矩阵 $\boldsymbol{M}$ 仅受执行体的历史信息影响，即执行体被攻击次数。而在实际情况中，攻击者是无法预先探知拟态系统中被调度执行体的种类的，即攻击者的攻击意图与调度算法无关。因此，矩阵 $\boldsymbol{M}$ 对不同算法的安全性影响是相同的，而

又由于增加了历史信息的影响，算法的动态性得到了提升（通过攻击者的攻击倾向而动态调度执行体）。

由此可见，HFAWA 在调度时会尽可能选择使系统共模漏洞数量少的执行体进行调度，能够在保证安全性的前提下表现出良好的动态性。

2 算法及实验改进

在碰撞实验中给出高阶异构度计算式 $H^{(n)}=P^{(n)}-S^{(n)}$，这样做的目的是尽量增大防御面积，选择防御面积最大的执行体集进行调度。但在实际的拟态场景中并非防御面积越大效果越好。在经典拟态场景中通常需要构建执行体池，由调度算法调度执行体进入运行池，最后根据裁决算法对执行体输出结果进行裁决。因此，一个优秀的调度算法往往需要根据拟态构造的裁决机制进行相应的调整。

目前，学者已提出了很多裁决算法，如基于高阶异构度的裁决算法[19]、基于二进制文件的裁决算法[26]、基于异常值的裁决算法[27]、大数裁决算法[28]等。相较于其他算法，大数裁决算法以其通用性被更广泛地应用于现有的 DHR 系统中，其基本思想是将执行体输出相同数量最多的结果作为判决结果。因此本节将以大数裁决算法为例，对 HFAWA 进行调整，同时构建相应测试模型对算法有效性进行测试。

本文在算法 1 和文献[15]碰撞实验的基础上做如下调整。

1) 将碰撞实验中执行体由防御原子组成改为执行体由漏洞原子组成，每个执行体的漏洞原子个数为最大威胁数的 $\frac{1}{10}$，且最小不少于 10 个。

2) 威胁生成策略不变，为随机装载和全装载。碰撞测试调整为若执行体中存在威胁集中的漏洞原子则被攻破。在 5 模异构执行体集中，若攻破执行体数超过 3 个，则判断系统被攻破。

大数裁决下 5 模执行体高阶异构度如表 4 所示。调整后的实验算法如算法 2 所示。

算法 2 大数裁决系统攻击实验算法

输入 系统执行体池 $\boldsymbol{\Omega}$，执行体调度概率矩阵 $\boldsymbol{M}$，调度总轮次 n，历史威胁记录矩阵 **HS**

输出 系统平均攻破率 P_A

1) 计算系统高阶异构度矩阵 $\boldsymbol{H}$，计算式如表 4 所示

表 4 大数裁决下 5 模执行体高阶异构度

高阶异构度	计算式
$H^{(1)}$	0
$H^{(2)}$	$-\left(\sum S^{(2)}\right)$
$H^{(3)}$	$-\left(\sum S^{(2)}-2\sum S^{(3)}\right)$
$H^{(4)}$	$-\left(\sum S^{(3)}-3\sum S^{(4)}\right)$
$H^{(5)}$	$-\left(\sum S^{(3)}-3\sum S^{(4)}+6\sum S^{(5)}\right)$

2) for j in $\{1,2,3,\cdots,n\}$
3) 根据威胁选取策略构造威胁矩阵 $\boldsymbol{T}$
4) for A_{W} in $\boldsymbol{\Omega}_{\mathrm{W}}$
5) for 漏洞原子 l_i in A_{W}
6) $\boldsymbol{M}[i]=\boldsymbol{M}[i]-\mathbf{HS}[i]$
7) end for
8) end for
9) 执行算法 1，得到运行池 $\boldsymbol{\Omega}_{\mathrm{R}}$
10) for A_{R} in $\boldsymbol{\Omega}_{\mathrm{R}}$
11) 对比 A_{R} 和 $\boldsymbol{T}$ 中各原子碰撞次数 c_i。攻破次数 $c_o=0$
12) if $c_i \geqslant \frac{N_{\mathrm{R}}}{2}$
13) $c_o=c_o+1$
14) end if
15) $P_j=\frac{c_o}{\text{威胁总数}}$
16) end for
17) end for
18) $P_A=\frac{\mathrm{sum}P_j}{n}$
19) 输出 P_A

算法 2 对应的实验流程如图 4 所示。

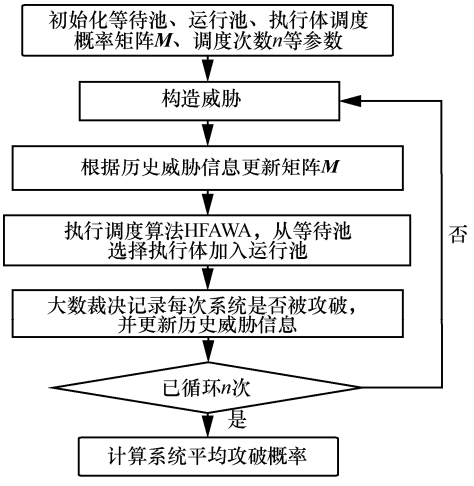


图 4 算法 2 对应的实验流程

3　失效率分析

采用大数裁决算法的拟态防御系统被攻破的概率等效于运行池中半数以上执行体产生共模漏洞的概率之和，因此可以采用被选中的异构执行体共模漏洞的面积与公共漏洞的面积之比作为系统被攻破的概率。

几个基本假设如下。

假设 1　攻击者每次随机使用一种攻击方式进行攻击，若该攻击方式能同时攻破半数以上处于运行池中的执行体，则拟态系统被攻破。

假设 2　HFAWA、FAWA、H2 算法[15]均使用执行体的历史信息作为部分调度依据，虽然具体使用方法有所差异，但为了便于分析，均使用 $P(\mathrm{HI})$ 表示被执行体历史信息（ HI, history information）所影响的执行体调度概率。

假设 3　当前等待池中执行体的漏洞数量为 $\{N_1, N_2, \cdots, N_z\}$，二阶共模漏洞数量为 $N_i^{(2)} = N_a \cap N_b$, $\cdots$,n 阶共模漏洞数量为 $N_i^{(n)} = N_a \cap N_b \cap \cdots \cap N_z$。漏洞总数为 $N_{\mathrm{sum}} = N_a \cup N_b \cup \cdots \cup N_z$。

根据以上假设，可得各算法在运行池大小为 k 的拟态系统中的失效率如表 5 所示。

表 5　各算法在运行池大小为 k 的拟态系统中的失效率

算法	失效率
HFAWA	$\dfrac{\sum_{i=1}^{\tau_1} N_i^{\left(\left\lceil \frac{k}{2} \right\rceil\right)}\Big\vert_{\min\left(N^{(k)}\left\vert_{\min\left(N^{(k-1)}\vert_{\min(\cdots)}+P(\mathrm{HI})\right)}+P(\mathrm{HI})\right.\right)}}{N_{\mathrm{sum}}}$
FAWA	$\dfrac{\sum_{i=1}^{\tau_1} N_i^{\left(\left\lceil \frac{k}{2} \right\rceil\right)}\Big\vert_{\min(P(\mathrm{HI}))}}{N_{\mathrm{sum}}}$
Ramdom, FIFO	$\dfrac{\sum_{i=1}^{\tau_1} N_i^{\left(\left\lceil \frac{k}{2} \right\rceil\right)}}{N_{\mathrm{sum}}}$
H2	$\dfrac{\sum_{i=1}^{\tau_1} N_i^{\left(\left\lceil \frac{k}{2} \right\rceil\right)}\Big\vert_{\min\left(\overline{N^{(2)}}+P(\mathrm{HI})\right)}}{N_{\mathrm{sum}}}$

接下来，将通过实验比较以上算法的安全性能。

4　实验验证

4.1　安全性实验

为了验证 HFAWA 的有效性，将其与其他 4 种调度算法（FAWA、FIFO、Random、H2）进行对比。其中 H2 为文献[16]中的算法，该文提出将异构度（2 阶）与历史信息（执行体运行次数）结合，其焦点在于裁决算法的改进，因此根据其思想实现“H2 调度算法”与本文算法进行对比分析。

本文通过 2 种实验对算法的安全性进行对比，实验分别为碰撞实验和大数裁决系统攻击实验。其中碰撞实验由文献[14]提出，其核心思想是通过对攻击原子和防御原子的碰撞率来衡量系统的安全性，碰撞率越高系统越安全，实验步骤如下。

Step1　等待池和运行池初始化。

Step2　构造威胁输入。

Step3　进行动态调度，从等待池中选取异构执行体填入运行池。

Step4　进行碰撞检测和计数。

Step5　返回 Step2，重复进行多次实验。

Step6　计算平均碰撞概率。

Step7　改变威胁数目，返回 Step1。

本文对碰撞实验进行改进，提出了 LSA，实验流程如图 4 所示。2 种实验的对比实验环境如表 6 所示。

表 6　对比实验环境

实验方式	装载方式	威胁构造
碰撞实验	部分装载	负反馈
LSA 实验	全装载	随机

表 6 中，随机部分装载与随机全装载均为随机装载，区别在于部分装载的执行体组成原子中可能会存在 0 原子（无效原子）；全装载的执行体组成原子均为非 0 原子（有效原子）。威胁构造方式分为负反馈和随机装载 2 种，区别在于随机装载为随机选取攻击原子进行装载，而负反馈装载会选取之前的系统攻击结果中攻破概率最高的攻击原子进行装载。

仿真中相关实验参数设定如下。

1) 执行体调度为 5 模异构执行体集，即每次从等待池中调度 5 个执行体到运行池中。

2) 执行体的防御/漏洞原子、威胁集均为随机生成。

3) 实验采用蒙特卡罗方法进行测试，每种实验环境下测试 10 000 次，最后取均值表示实验结果。

4) 负反馈威胁实验中威胁原子更新概率式与文献[15]中相同，为

$$\begin{cases} P_1 = \mathrm{e}^{-(P_1+p)} \\ P_2 = \min(1, \mathrm{e}^{-(P_2-p)}) \end{cases} \tag{10}$$

其中，P_1 和 P_2 在碰撞实验中分别表示发生碰撞和未发生碰撞的威胁原子被选中的概率，在 LSA 实验中分别表示未攻破系统和攻破系统的威胁原子被选中的概率；p 为单次实验中该威胁原子与防御/漏洞原子发生碰撞的概率。

实验参数配置如表 7 所示。

表 7　　实验参数配置

实验方式	运行池大小	等待池大小	异构执行体容量
碰撞实验	5	20	10
LSA 实验	5	20	$\begin{cases} 10，威胁数 \leqslant 100 \\ \dfrac{威胁数}{10}，威胁数 > 100 \end{cases}$

本文实验假设每次从等待池中调度 5 个执行体（其中原子种类随机生成）进入运行池中工作，因此设定运行池大小为 5，等待池大小为 20。在碰撞实验中异构执行体容量参考文献[15]中参数进行设置；在 LSA 实验中由于执行体中原子为漏洞原子，在通常漏洞发现的过程中，单一类型系统/软件存在的漏洞数量应与同功能的系统/软件发现的漏洞数量总数存在比例关系。因此在 LSA 实验中，当威胁数大于 100 时异构执行体容量设定为 $\dfrac{威胁数}{10}$，其执行体容量变化如图 5 所示。实验结构如图 6 所示。

实验程序用 Python 编写。为更好地量化实验结果，碰撞实验中用碰撞率表示威胁被系统防御的概率，碰撞率越高，代表系统防御能力越强；LSA 实验中用系统攻破率表示系统被威胁攻破的概率，系统攻破率越低，代表系统越安全。

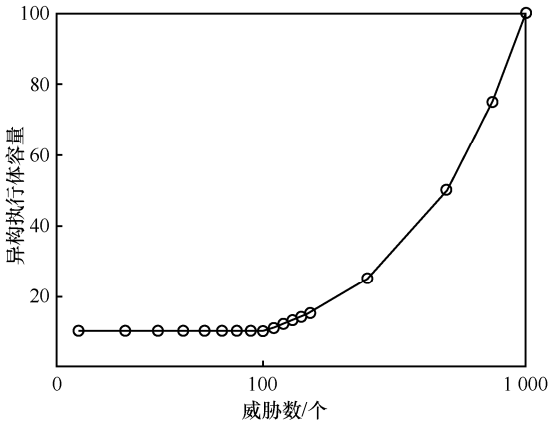


图 5　LSA 实验执行体容量变化

4.1.1　碰撞实验

根据表 5 中的环境描述，在碰撞实验中分别测试了执行体负反馈威胁下部分装载、负反馈威胁下全装载、随机威胁下部分装载和随机威胁下全装载 4 种环境，得到的实验结果如图 7 和图 8 所示。

从图 7 和图 8 的实验结果可以看出，在全装载和部分装载的环境下，HFAWA 都具有优于其他算法的防御效果，且这种优势在威胁数较少时更明显。而攻击方式的改变会对系统的安全性产生一定威胁，但这种威胁主要体现在威胁数较少时碰撞率会有些许降低，威胁数较大时负反馈攻击的碰撞率反而高于随机攻击的碰撞率，这与文献[15]中的结论是一致的。

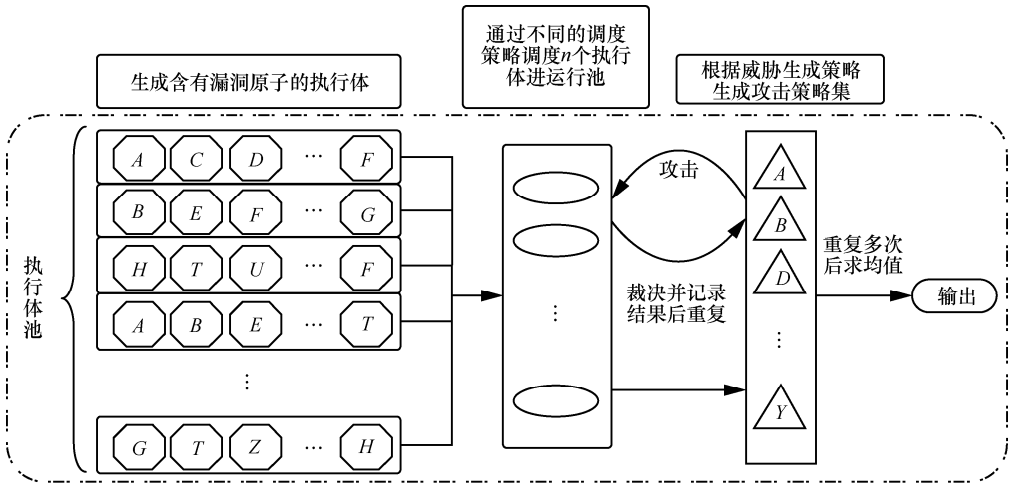
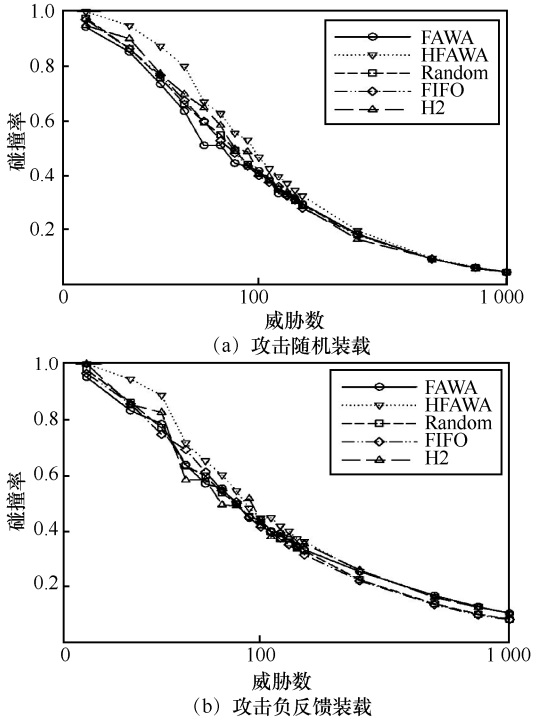


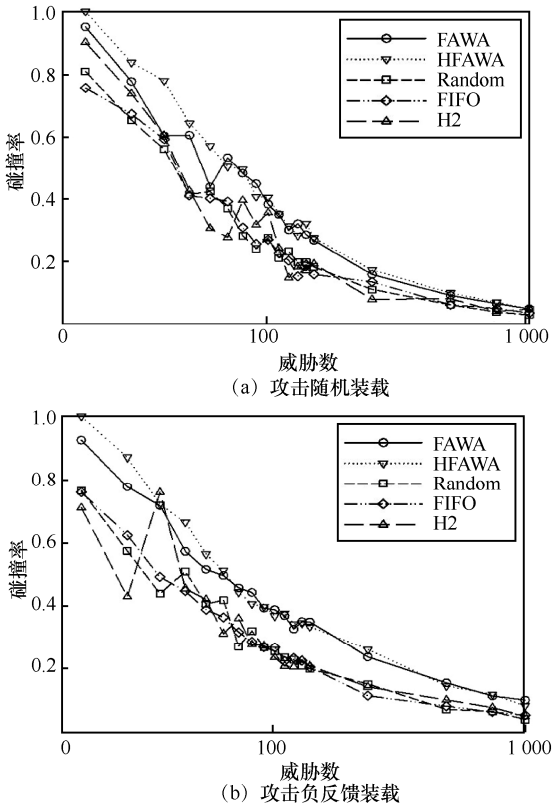
图 6　实验结构



（a）攻击随机装载

（b）攻击负反馈装载

图 7　碰撞实验下执行体全装载



（a）攻击随机装载

（b）攻击负反馈装载

图 8　碰撞实验下执行体部分装载

由于在实际场景中，异构执行体间虽然拥有相同的层级数，但其具有的防御能力和漏洞类型/个数往往都是不同的，因此部分装载的实验环境更能体现实际场景中的效果。给出部分装载环境下的详细实验数据，如表 8 和表 9 所示。

表 8　随机攻击碰撞试验下执行体部分装载实验结果

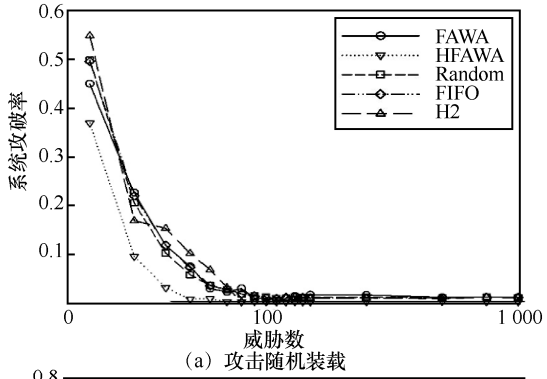
威胁个数	FAWA	HFAWA	Random	FIFO	H2
20	0.950 8	1.000 0	0.806 4	0.753 8	0.900 7
30	0.774 1	0.835 5	0.649 8	0.670 2	0.735 0
40	0.599 2	0.776 5	0.555 4	0.588 1	0.600 8
50	0.601 4	0.639 7	0.408 5	0.404 2	0.420 7
60	0.432 6	0.566 2	0.417 2	0.395 2	0.299 1
70	0.528 5	0.498 4	0.363 4	0.386 1	0.270 4
80	0.476 9	0.489 2	0.275 7	0.301 8	0.389 5
90	0.442 7	0.401 0	0.234 2	0.249 6	0.311 0
100	0.376 5	0.399 0	0.269 7	0.262 4	0.349 8
110	0.343 0	0.346 9	0.205 4	0.220 9	0.235 6
120	0.294 0	0.306 7	0.225 5	0.196 9	0.141 5
130	0.314 1	0.276 0	0.191 8	0.146 4	0.177 4
140	0.278 7	0.314 9	0.191 5	0.184 1	0.164 1
150	0.260 6	0.268 0	0.176 3	0.152 0	0.186 6
250	0.152 3	0.166 7	0.104 0	0.128 5	0.072 0
500	0.084 4	0.092 7	0.054 7	0.055 0	0.074 3
750	0.058 9	0.062 4	0.031 0	0.043 2	0.033 3
1 000	0.042 1	0.041 9	0.022 0	0.027 8	0.040 8

表 9　负反馈攻击碰撞试验下执行体部分装载实验结果

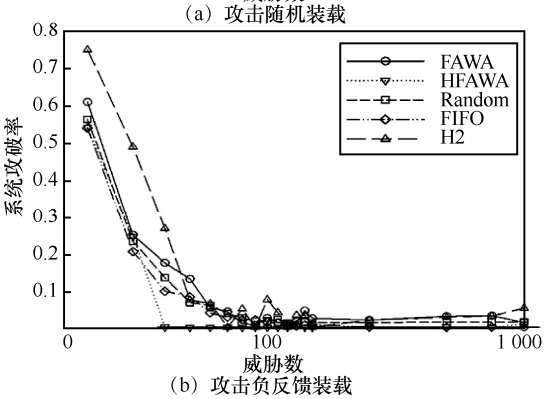
威胁个数	FAWA	HFAWA	Random	FIFO	H2
20	0.924 9	0.999 9	0.765 2	0.759 9	0.710 2
30	0.777 0	0.870 6	0.573 6	0.623 6	0.430 1
40	0.716 7	0.721 1	0.439 7	0.492 4	0.758 7
50	0.572 8	0.663 6	0.510 2	0.448 4	0.456 3
60	0.516 8	0.563 9	0.407 1	0.388 6	0.421 1
70	0.497 0	0.513 1	0.419 1	0.364 8	0.312 7
80	0.457 8	0.444 0	0.272 1	0.317 8	0.360 5
90	0.443 2	0.407 3	0.321 2	0.286 4	0.277 5
100	0.393 8	0.398 3	0.269 3	0.269 4	0.274 9
110	0.388 9	0.368 7	0.257 6	0.268 8	0.238 4
120	0.369 8	0.377 6	0.238 2	0.225 9	0.210 4
130	0.327 3	0.343 4	0.230 0	0.235 7	0.207 8
140	0.350 8	0.346 0	0.212 3	0.227 9	0.229 4
150	0.350 4	0.335 2	0.201 7	0.206 8	0.210 8
250	0.239 1	0.262 3	0.153 6	0.117 0	0.144 8
500	0.157 4	0.147 5	0.074 1	0.083 9	0.103 5
750	0.116 3	0.121 0	0.067 2	0.065 6	0.079 8
1 000	0.103 1	0.085 2	0.042 4	0.055 6	0.052 0

4.1.2　LSA 实验

根据图 4 所示流程进行实验，LSA 实验中同样测试了表 5 中描述的 4 种环境，得到的实验结果如图 9 和图 10 所示。

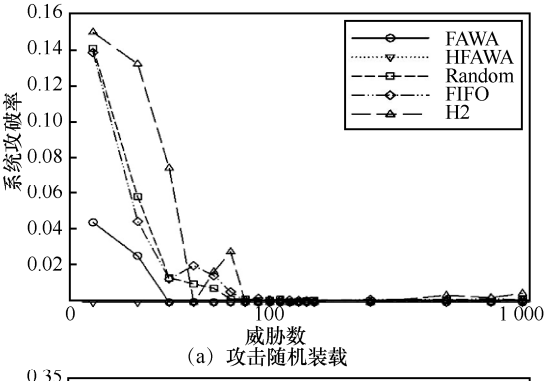


（a）攻击随机装载

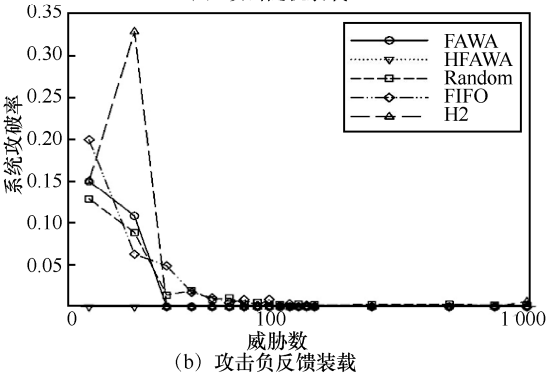


（b）攻击负反馈装载

图 9　LSA 实验下执行体全装载



（a）攻击随机装载



（b）攻击负反馈装载

图 10　LSA 实验下执行体部分装载

从图 9 和图 10 的实验结果可以看出，在 LSA 实验中，HFAWA 依然保持着良好的防御能力。虽然在攻击方式由随机攻击变为负反馈攻击后，系统攻破率会有些许上升，但通过对比可以看出，受影响较大的是威胁数较少的情况（H2 算法受到的影响最大），随着威胁数增大，攻击方式的改变对系统攻破率的影响可以忽略。给出部分装载实验的详细数据，如表 10 和表 11 所示。

表 10　随机攻击大数裁决下执行体部分装载实验结果

威胁个数	FAWA	HFAWA	Random	FIFO	H2
20	0.044 2	0	0.141 6	0.139 1	0.150 4
30	0.025 7	0	0.058 3	0.044 8	0.132 8
40	0	0	0.013 6	0.012 9	0.074 9
50	0.000 2	0	0.010 2	0.020 3	0
60	0	0	0.007 8	0.014 8	0.016 9
70	0	0	0.001 8	0.005 8	0.028 0
80	0	0	0.001 8	0.001 0	0
90	0	0	0.001 0	0.002 3	0
100	0	0	0.001 3	0.001 3	0
110	0	0	0.001 6	0.001 2	0
120	0	0	0.000 4	0.001 3	0
130	0	0	0.000 6	0.001 1	0
140	0	0	0.000 9	0.000 8	0
150	0	0	0.001 3	0.000 6	0
250	0	0	0.000 8	0.001 7	0
500	0	0	0.000 8	0.001 4	0.003 9
750	0	0	0.000 6	0.002 1	0.002 6
1 000	0	0	0.001 4	0.001 2	0.005 0

表 11　负反馈攻击大数裁决下执行体部分装载实验结果

威胁个数	FAWA	HFAWA	Random	FIFO	H2
20	0.149 1	0	0.127 3	0.199 2	0.149 9
30	0.107 3	0	0.087 7	0.062 1	0.328 3
40	0	0	0.013 8	0.048 4	0
50	0	0	0.019 3	0.017 0	0
60	0	0	0.008 4	0.010 6	0
70	0	0	0.010 3	0.004 8	0
80	0	0	0.003 2	0.008 5	0
90	0	0	0.004 9	0.001 5	0
100	0	0	0.001 1	0.008 6	0
110	0	0	0.002 8	0.001 4	0
120	0	0	0.000 6	0.003 4	0
130	0	0	0.003 4	0.000 1	0
140	0	0	0.000 3	0.001 8	0
150	0	0	0.002 4	0.000 4	0
250	0	0	0.002 6	0	0
500	0	0	0.003 0	0	0
750	0	0	0.002 0	0	0
1 000	0	0	0.001 6	0	0.006 2

4.2　算法性能测试

根据以上实验参数配置，本文对 5 种调度算法的运行时间进行仿真实验，实验环境如表 12 所示。

表 12　　性能测试环境

条目	参数
CPU 型号	Intel(R) Core(TM)i5-7300HQ CPU@2.50 GHz
内存大小	8 GB
操作系统	Windows 10
运行语言	Python

实验只测试算法在调度选择过程的时间消耗，由于 HFAWA、H2 算法中异构度数组的构建为前期准备工作，因此不将其算作算法时间消耗内。实验测试每种算法调度 10 000 次的时间消耗，详细数据如表 13 所示。

表 13　　算法时间消耗

威胁个数	FAWA/s	HFAWA/s	Random/s	FIFO/s	H2/s
20	4.991 6	7.872 9	1.686 5	4.616 7	17.165 1
30	5.391 6	8.100 3	1.811 2	4.893 9	17.824 3
40	5.820 4	8.576 1	2.012 6	4.969 7	17.916 1
50	6.039 8	8.892 2	2.319 8	5.171 2	17.951 0
60	6.288 2	9.019 9	2.453 4	5.408 5	18.414 7
70	6.695 1	9.293 1	2.744 7	5.659 9	17.980 9
80	7.223 6	9.586 4	2.962 1	5.894 2	17.858 2
90	7.437 1	9.804 8	3.159 5	6.189 4	18.202 3
100	7.745 3	10.276 5	3.678 2	6.557 5	18.321 0
110	8.124 3	10.505 9	3.798 8	6.661 2	18.705 4
120	8.596 0	11.144 2	3.931 5	6.599 4	19.000 2
130	9.309 1	11.283 8	4.283 5	6.862 6	19.165 8
140	9.093 7	11.765 5	4.255 6	7.167 3	19.285 4
150	9.497 6	12.495 6	4.546 8	7.377 3	19.297 4
250	12.747 9	15.552 4	6.559 5	9.276 2	21.753 8
500	21.969 2	23.834 3	12.948 4	15.232 3	27.765 7
750	29.718 5	31.714 2	18.338 0	20.575 0	33.197 2
1 000	38.137 0	39.749 8	25.672 3	26.013 4	38.853 1

在笛卡尔坐标系中绘制表 13 所示实验数据，如图 11 所示。

根据以上实验结果，可以得到以下结论。

1) 在碰撞实验中，HFAWA 的碰撞率基本上高于其他算法。当威胁数较小时，HFAWA 优势较明显；当威胁数大时，HFAWA 和 FAWA 的碰撞率相近且高于其他算法。这是因为威胁数越大，异构体选择防御原子相同的概率就越小。当威胁数趋近于无穷时，异构体间出现相同防御原子的概率为 0，这时 HFAWA 与 FAWA 效果相同。

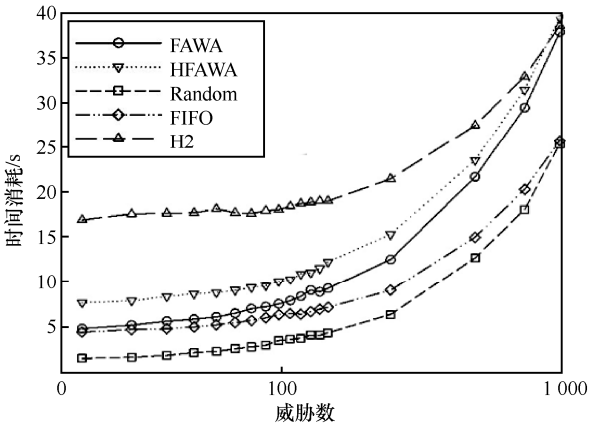


图 11　算法时间消耗

2) 在 LSA 实验中，HFAWA 的系统攻破率始终低于其他算法。在异构体部分装载环境下，使用 HFAWA 的系统被攻破概率为 0。这是因为在异构体部分装载环境下，5 模执行体中极易存在 3 模相似度为 0（异构度最大）的情况，这时根据高阶异构度调度执行体的 HFAWA 效果明显优于其他算法。

3) 由算法性能测试结果可明显看出，HFAWA 在威胁数少的情况下相较于其他 3 种算法运行速度较慢，而较 H2 算法更快。这是由于 HFAWA 的时间消耗主要是算法中异构度数组的比较，这与运行池的大小有直接关系，与威胁个数关系不大，当威胁个数较大时这一消耗可以忽略。而相较于 H2 算法中每次实验都需要更新的历史信息为二维数组，HFAWA 仅需对一维数组进行更新，因此 HFAWA 具有优于 H2 算法的运行效率。

将以上实验算法的防御能力和运行速度指标进行对比，结果如表 14 所示。

表 14　　算法指标对比

算法	防御能力	运行速度
HFAWA	高	中
FAWA	较高	中
H2	中	较慢
FIFO	较低	较快
Random	较低	快

综上，HFAWA 能够在保证系统安全的同时，不造成过高的系统开销。

5　结束语

针对目前 DHR 系统对异构度的考虑仅局限于二阶层面，造成系统安全性不足且调度异构体缺乏动态性的问题。本文提出了 HFAWA 调度算法，并通过对容斥原理进行扩展应用，给出了高阶异构度的计算方式。仿真实验验证了 HFAWA 具有良好的防御能力和运行速度。后续将基于本文所做工作，继续优化执行体的调度算法，进一步提高系统安全性。

参考文献：

[1] 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.

[2] 石乐义, 郭宏彬, 温晓, 等. 端信息跳扩混合的主动网络防御技术研究[J]. 通信学报, 2019, 40(5): 125-135.
SHI L Y, GUO H B, WEN X, et al. Research on end hopping and spreading for active cyber defense[J]. Journal on Communications, 2019, 40(5): 125-135.

[3] 何永忠, 陈美玲. 基于协议的拟态研究综述[J]. 北京交通大学学报, 2016, 40(5): 1-8.
HE Y Z, CHEN M L. Protocol mimicry technique and its development[J]. Journal of Beijing Jiaotong University, 2016, 40(5): 1-8.

[4] VANO-GARCIA F, MARCO-GISBERT H. KASLR-MT: kernel address space layout randomization for multi-tenant cloud systems[J]. Journal of Parallel and Distributed Computing, 2020, 137: 77-90.

[5] 马多贺, 李琼, 林东岱. 基于 POF 的网络窃听攻击移动目标防御方法[J]. 通信学报, 2018, 39(2): 73-87.
MA D H, LI Q, LIN D D. Moving target defense against network eavesdropping attack using POF[J]. Journal on Communications, 2018, 39(2): 73-87.

[6] KRYLOV V, KRAVTSOV K. Principles of network security protocols based on dynamic address space randomization[J]. Journal of Communication and Computer, 2016, 13(2): 77-89.

[7] FENSKE E, BROWN D, MARTIN J, et al. Three years later: a study of MAC address randomization in mobile devices and when it succeeds[J]. Proceedings on Privacy Enhancing Technologies, 2021, 2021(3): 164-181.

[8] 马博林, 张铮, 陈源, 等. 基于指令集随机化的抗代码注入攻击方法[J]. 信息安全学报, 2020, 5(4): 30-43.
MA B L, ZHANG Z, CHEN Y, et al. The defense method for code-injection attacks based on instruction set randomization[J]. Journal of Cyber Security, 2020, 5(4): 30-43.

[9] 郝志宇, 翟健宏, 云晓春, 等. 动态路由模拟策略研究[J]. 通信学报, 2007, 28(12): 19-24.
HAO Z Y, ZHAI J H, YUN X C, et al. Research on dynamic routing mechanism in network simulation[J]. Journal on Communications, 2007, 28(12): 19-24.

[10] 邬江兴. 网络空间拟态安全防御[J]. 保密科学技术, 2014(10): 4-9, 1.
WU J X. Mimic security defense in cyberspace[J]. Secrecy Science and Technology, 2014(10): 4-9, 1.

[11] 姜远海. 基于分离映射的网络层主动变迁技术研究[D]. 北京: 北京交通大学, 2016.
JIANG Y H. Research on active change technology of network layer based on separation mapping[D]. Beijing: Beijing Jiaotong University, 2016.

[12] 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.

[13] 吴铤, 胡程楠, 陈庆南, 等. 基于执行体划分的防御增强型动态异构冗余架构[J]. 通信学报, 2021, 42(3): 122-134.
WU T, HU C N, CHEN Q N, et al. Defense-enhanced dynamic heterogeneous redundancy architecture based on executor partition[J]. Journal on Communications, 2021, 42(3): 122-134.

[14] 朱正彬, 刘勤让, 刘冬培, 等. 拟态多执行体调度算法研究进展[J]. 通信学报, 2021, 42(5): 179-190.
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.

[15] 杨林, 王永杰, 张俊. FAWA: 一种异构执行体的负反馈动态调度算法[J]. 计算机科学, 2021, 48(8): 284-290.
YANG L, WANG Y J, ZHANG J. FAWA: a negative feedback dynamic scheduling algorithm for heterogeneous executor[J]. Computer Science, 2021, 48(8): 284-290.

[16] 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, 2020, 46(5): 12-18.
WU Z Q, ZHANG F, GUO W, et al. A mimic arbitration optimization method based on heterogeneous degree of executors[J]. Computer Engineering, 2020, 46(5): 12-18.

[17] 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.

[18] 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.

[19] 魏帅, 张辉华, 苏野, 等. 面向拟态防御系统的高阶异构度大数判决算法[J]. 计算机工程, 2021, 47(5): 30-35.

WEI S, ZHANG H H, SU Y, et al. Majority voting algorithm based on high-order heterogeneity for mimic defense system[J]. Computer Engineering, 2021, 47(5): 30-35.

[20] 杨欣欣, 黄少滨. 高阶异构数据层次联合聚类算法[J]. 计算机研究与发展, 2015, 52(1): 200-210.
YANG X X, HUANG S B. A hierarchical co-clustering algorithm for high-order heterogeneous data[J]. Journal of Computer Research and Development, 2015, 52(1): 200-210.

[21] 李超. 高阶多数据集建模新方法与应用研究[D]. 哈尔滨: 哈尔滨工程大学, 2017.
LI C. Study on novel modeling methods and applications for multiple higher order datasets[D]. Harbin: Harbin Engineering University, 2017.

[22] 黄少滨, 杨欣欣, 申林山, 等. 高阶异构数据模糊联合聚类算法[J]. 通信学报, 2014, 35(6): 15-24.
HUANG S B, YANG X X, SHEN L S, et al. Fuzzy co-clustering algorithm for high-order heterogeneous data[J]. Journal on Communications, 2014, 35(6): 15-24.

[23] 丁绍虎, 齐宁, 郭义伟. 基于 M-FlipIt 博弈模型的拟态防御策略评估[J]. 通信学报, 2020, 41(7): 186-194.
DING S H, QI N, GUO Y W. Evaluation of mimic defense strategy based on M-FlipIt game model[J]. Journal on Communications, 2020, 41(7): 186-194.

[24] 蔡雨彤, 常晓林, 石禹, 等. 动态平台技术防御攻击的瞬态效能量化分析[J]. 信息安全学报, 2019, 4(4): 59-67.
CAI Y T, CHANG X L, SHI Y, et al. Analyzing transient effectiveness of dynamic platform technique in resisting attacks[J]. Journal of Cyber Security, 2019, 4(4): 59-67.

[25] 张兴明, 顾泽宇, 魏帅, 等. 拟态防御马尔可夫博弈模型及防御策略选择[J]. 通信学报, 2018, 39(10): 143-154.
ZHANG X M, GU Z Y, WEI S, et al. Markov game modeling of mimic defense and defense strategy determination[J]. Journal on Communications, 2018, 39(10): 143-154.

[26] 吴正江, 姚琪, 冯四风, 等. 基于数据库二进制日志的竞赛式仲裁优化方案[J]. 计算机工程, 2021, 47(5): 24-29.
WU Z J, YAO Q, FENG S F, et al. Optimization scheme of competitive arbitration based on binary database log[J]. Computer Engineering, 2021, 47(5): 24-29.

[27] 高振斌, 贾广瑞, 张文建, 等. 基于异常值的拟态裁决优化方法[J]. 计算机应用研究, 2021, 38(7): 2066-2071.
GAO Z B, JIA G R, ZHANG W J, et al. Mimic ruling optimization method based on executive outliers[J]. Application Research of Computers, 2021, 38(7): 2066-2071.

[28] CHOI J, GOH K I. Dynamics of consensus formation on multiplex networks: the majority-vote model[C]//APS March Meeting. Washington, D.C.: American Physical Society, 2018:1-15.

[作者简介]

贾洪勇（1975− ），男，河南西平人，博士，郑州大学讲师，主要研究方向为云计算安全和拟态防御。

潘云飞（1998− ），男，河南郑州人，郑州大学硕士生，主要研究方向为网络空间安全和拟态防御。

刘文贺（1999− ），男，河南周口人，郑州大学硕士生，主要研究方向为网络空间安全和拟态防御。

曾俊杰（1977− ），男，江西南康人，郑州大学讲师，主要研究方向为网络与信息安全。

张建辉（1977− ），男，河南平顶山人，博士，国家数字交换系统工程技术研究中心副研究员，主要研究方向为网络与信息安全、网络体系架构、人工智能。