基于执行体划分的防御增强型动态异构冗余架构

吴铤[1,2]，胡程楠[1]，陈庆南[1]，陈安邦[1]，郑秋华[1]

（1. 杭州电子科技大学网络空间安全学院，浙江 杭州 310018；2. 北京航空航天大学杭州创新研究院，浙江 杭州 310051）

摘 要：针对 DHR 系统服务体在面临共同漏洞时的系统脆弱性问题，提出了一种改进的 DHR 架构——IDHR。该架构在 DHR 的基础上，首先引入根据执行体间的异构性对执行体集进行划分的执行体划分模块，以极大增强各执行体池之间的异构性。在此基础上，改进调度模块中的动态选择算法，即采用先随机选择执行体池，再从执行体池中随机选择执行体的方式，以提高在共同漏洞下 DHR 系统的安全性。最后，通过随机模拟执行体和仿真 Web 服务器 2 种实验方案，从攻击成功率和被控制率 2 个方面对所提 IDHR 架构进行安全性评估。实验结果表明，IDHR 架构的安全性，尤其是在共同漏洞未知情况下，明显优于传统 DHR 架构。

关键词：拟态防御；拟态系统架构；动态异构冗余；安全性分析

中图分类号：TP309.1

文献标识码：A

DOI: 10.11959/j.issn.1000−436x.2021022

Defense-enhanced dynamic heterogeneous redundancy architecture based on executor partition

WU Ting[1,2], HU Chengnan[1], CHEN Qingnan[1], CHEN Anbang[1], ZHENG Qiuhua[1]

1. School of Cyberspace Security, Hangzhou Dianzi University, Hangzhou 310018, China
2. Hangzhou Innovation Institute, Beihang University, Hangzhou 310051, China

Abstract: Aiming at the security problem when servants are faced with common vulnerabilities, an improved DHR architecture called IDHR was proposed. On the basis of DHR, an executor-partition module that divided the executor-set to several executor pools by the heterogeneity among the executors was introduced to improve the heterogeneity among the executor pools. Moreover, the scheduling algorithm was improved by choosing executor pools randomly at first, and then choosing the executors from these pools randomly. Finally, through two experimental schemes of random simulation and Web server emulation, the security evaluation of the proposed IDHR architecture was carried out from two aspects of attack success rate and control rate. Experimental results show that the security of the IDHR architecture, especially when the common vulnerability is unknown, is significantly better than the traditional DHR architecture.

Keywords: mimic defense, mimic system architecture, dynamic heterogeneous redundancy, security analysis

1 引言

随着信息技术的飞速发展，网络空间安全问题日益突出。一方面，现有大多数网络信息系统基本采用静态体系结构，这使其很容易受到诸如 0day 攻击等新型网络攻击的威胁。另一方面，由于软硬件的供应链组成复杂，且其与软件开发过程中的人员、工具、环境等因素密切相关，系统中存在安全漏洞和后门几乎是不可避免的，并且维护者很难识别和修复所存在的安全漏洞和后门[1]。在面临以上问题时，传统的被动式静态防御系统（如防火墙、入侵检测技术等）无法确保系统的安全性[2]。因此，

收稿日期：2020−10−20；**修回日期**：2020−12−30

通信作者：郑秋华，zqh@hdu.edu.cn

基金项目：浙江省重点研发计划基金资助项目（No.2020C01078, No.2019C01012, No.2017C01062）

Foundation Item: Zhejiang Province Key Research and Development Program (No.2020C01078, No.2019C01012, No.2017C01062)

当今网络空间安全的一个根本挑战是如何设计新的系统防御机制，使其在系统存在安全漏洞的情况下，仍然可以保障系统的安全性和可靠性。

针对此问题，美国国家科学与技术委员会（NSTC, National Science and Technology Council）提出了一种典型的动态主动防御技术——移动目标防御（MTD, moving target defense）技术[3]。MTD 的基本思想是通过动态的、多样化的系统构建与部署策略，增大攻击者对系统架构分析探测的成本，实现提高系统安全性的目的。近几年来，MTD 系统在网络安全领域中各个研究场景发展迅速，其安全性相较于被动式静态防御系统有显著提升[4-5]。然而，MTD 系统在本质上可看作单模动态异构冗余系统，其有限的动态变化路径容易被攻击者穷举，进而使系统被攻破。针对以上问题，2014 年邬江兴[6-7]提出了“拟态防御”思想，即在功能等价条件下，通过网络、平台、环境、软件等多模主动跳变或快速迁移来实现拟态环境，使攻击者难以观察和预测目标的变化，从而大幅增加其攻击难度和成本。拟态防御的典型架构是动态异构冗余（DHR, dynamic heterogeneous redundancy）架构[8]。DHR 架构的安全性依赖于系统的动态性、随机性和异构性，其中异构性是决定 DHR 架构安全性的关键，其很大程度上决定了 DHR 系统的安全上限[9]。理论上，当一个 DHR 系统中各执行体不存在共同漏洞时，该系统是足够安全的[10-11]。然而在实践中，DHR 系统中各执行体通常会存在一些相同漏洞。当服务体中包含同一漏洞的执行体数大于系统表决模时，攻击者就可利用该共同漏洞对系统进行攻击，并成功绕过表决器，从而达到攻击成功的目的[9]。虽然当服务体共同漏洞已知时，可通过修补机制解决服务体漏洞所导致的安全性下降问题，但是当共同漏洞未知时，DHR 系统脆弱性问题依然存在。

为此，本文提出了一种改进的 DHR 架构——IDHR（improved dynamic heterogeneous redundancy）架构。该架构在 DHR 架构基础上，引入了一个执行体划分模块，同时对调度模块中的动态选择算法进行了改进，即在 IDHR 架构中，执行体划分模块根据执行体间的异构性，将执行体划分到多个彼此异构性较大的执行体子池。当 IDHR 系统运行时，调度模块随机选择若干个执行体子池，然后从子池中随机选择执行体。上述 2 个改进可有效降低所选择服务体包含共同漏洞的可能性，进而提高系统的安全性。

本文的主要贡献如下。

1) 通过引入执行体划分模块和改进调度模块中的动态选择算法，对 DHR 架构进行改进，达到了在服务体构建过程中尽量减少存在共同漏洞的服务体的目的，缓解因共同漏洞导致的 DHR 系统脆弱性问题。

2) 基于聚类思想提出了一种执行体集划分算法，解决了如何将 DHR 架构中的执行体划分到多个彼此异构性较大的执行体子池的问题。

3) 利用基于概率的 DHR 安全性分析方法对 IDHR 模型进行了安全性分析，给出了在合谋盲攻击策略下 IDHR 系统攻击成功率和被控制率计算方法，解决了 IDHR 系统的安全性量化评估问题。

4) 通过随机模拟执行体和仿真 Web 服务器 2 种实验方案对 IDHR 架构安全性进行评估。验证了 IDHR 架构相较于 DHR 架构具有明显的安全性优势，尤其是在共同漏洞未知的情况下。

2　相关工作

学术界对 DHR 架构的研究大致分为以下 3 个方面：DHR 架构的应用研究、DHR 架构的安全性分析和 DHR 架构的优化与完善。

在 DHR 架构的应用研究中，仝青等[12]和张铮等[13]基于 DHR 架构构建了拟态防御 Web 服务器，并完成安全性和性能测试，验证了拟态防御技术在 Web 服务器上的有效性和可行性。宋克等[14]和马海龙等[15]针对以太网交换机和路由器面临未知漏洞和未知后门时的安全威胁，提出了相应基于 DHR 的交换机与路由器内生安全架构，实现了交换机及路由器系统的主动防御。丁绍虎等[16]给出了基于 DHR 架构的软件定义网络（SDN, software defined network）控制层安全机制，解决了 SDN 控制层的单点脆弱性问题。周清雷等[17]基于 DHR 架构，提出了以移动端二维码为接口、以动态口令为内核的物理访问控制拟态防御认证方法，解决了传统物理访问控制系统的认证方法易受攻击的安全问题。

在 DHR 架构的安全性分析中，任权等[18]利用离散时间马尔可夫链模型对 DHR 系统进行建模，并对 DHR 抗干扰性能进行分析，得到目标系统在攻击扰动条件下的稳态可用性和感知安全性。朱维军等[19]基于并行自动机与交替自动机，提出拟态防御自动机，把 DHR 安全性自动分析问题规约

为交替自动机模型检测问题，完成系统的安全性强弱分析。

在 DHR 架构的优化与完善领域，Zhang 等[20]基于 DHR 架构提出安全本体的概念，给出适用于拟态防御的系统建模方法，解决了拟态防御系统中无法使用传统安全建模方法的问题。李卫超等[21]提出基于防御能力、运行效率和系统恢复 3 个方面的多模表决建模分析方法，得到模型变化趋势，给出系统实际部署建议，克服了 DHR 架构中因使用多模表决方法不当而导致的系统安全性下降问题。

由上可见，目前针对 DHR 架构的优化均没有考虑到当系统存在共同漏洞时的系统脆弱性问题。

3 DHR 模型介绍和存在问题分析

DHR 系统架构[8]如图 1 所示，其主要由 5 个部分组成，具体介绍如下。

1) 输入模块

此模块包括用户输入和输入代理。输入代理将用户输入复制 n 份，形成 n 个子输入后，分发给处理模块中的服务体 S。

2) 处理模块

此模块由服务体 S 组成，S 中的各个执行体 $\{A_1, A_2, \cdots, A_n\}$ 对输入模块形成的 n 个子输入进行处理，并形成 n 个子输出发送至表决器，其中参数 n 为选择模。

3) 输出模块

此模块由 k 模表决器组成，若在 n 个子输出中，存在 k 个或 k 个以上结果一致，则输出该结果，否则阻断输出，其中参数 k 为表决模。

4) 构建模块

构建模块包括由若干个构件组所构成的构件集合 C 和执行体集 E。根据系统构建规则，从每个构件组中选择某一构件 FC，得到系统执行体 E_i，进而得到系统执行体集 $E=\{E_1, \cdots, E_i, \cdots, E_m\}$，其中 $m(m>n)$ 为执行体总数。

5) 调度模块

该模块中运行动态选择算法 RS，即根据随机参数，从执行体集 E 中随机选择 n 个执行体，作为处理模块中的服务体 S。在调度周期 t_s 后，调度模块将服务体 S 下线，通过清洗还原到初始状态，并根据随机参数重新生成下一个调度周期对应的服务体。

以下通过一个案例，对 DHR 系统安全性进行简要分析。

例 1 设某个 DHR 系统的执行体集 $E=\{E_1, E_2, E_3, E_4, E_5, E_6\}$，系统漏洞集合 $V=\{v_1, v_2, v_3, v_4, v_5\}$，系统选择模 $n=3$，表决模 $k=2$。其中 $E_1, \cdots, E_6$ 所包含的漏洞分别为 $\{v_1, v_4, v_5\}$、$\{v_1, v_2\}$、$\{v_3, v_4\}$、$\{v_2, v_5\}$、$\{v_1, v_3, v_5\}$ 和 $\{v_2, v_4\}$。

由于 DHR 从执行体集中任选 $n=3$ 个执行体作为一个服务体，因此系统共包含 $C_6^3=20$ 个服务体，分别为

$$S_1=\{E_1, E_2, E_3\}, S_2=\{E_1, E_2, E_4,\}, S_3=\{E_1, E_2, E_5\},$$

$$\vdots$$

$$S_{18}=\{E_3, E_4, E_6\}, S_{19}=\{E_3, E_5, E_6,\}, S_{20}=\{E_4, E_5, E_6\}$$

记 $S=\{S_1, S_2, \cdots, S_{20}\}$。设不同执行体在遭到针

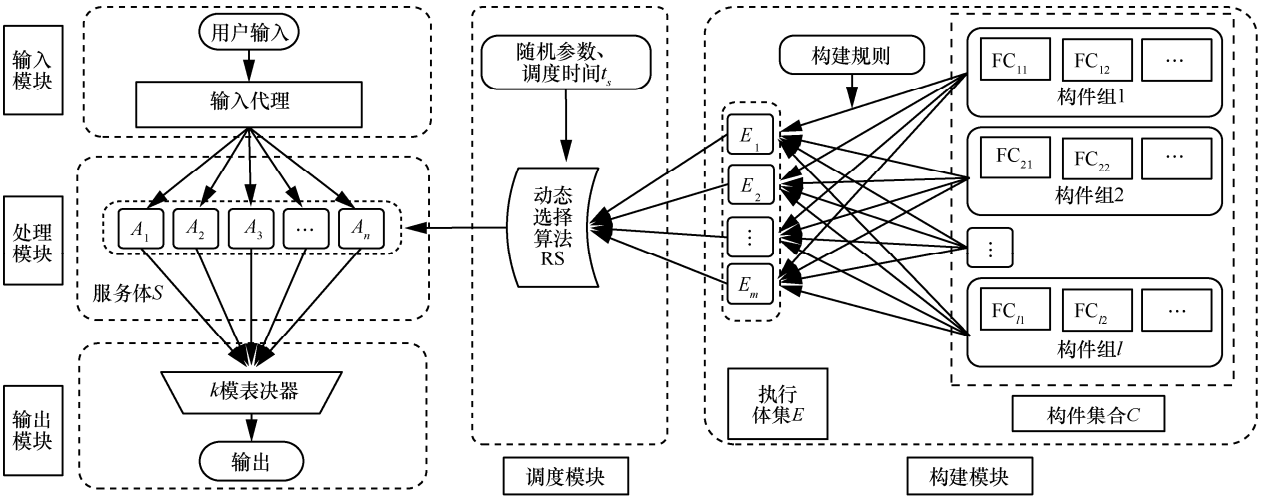

图 1 DHR 系统架构

对同一漏洞的攻击时，其输出保持一致[8]，则当调度模块所选择的服务体为 S_3 时，由于 S_3 中各个执行体所包含的漏洞分别为 $\{v_1,v_4,v_5\}$、$\{v_1,v_2\}$、$\{v_1,v_3,v_5\}$，若攻击者利用漏洞 v_3 发起攻击，则 DHR 处理模块中只有执行体 E_5 会产生异常输出，但因其数量小于系统表决模 2，系统将阻断该异常输出，攻击失败。当攻击者利用漏洞 v_1 进行攻击时，由于所有 3 个执行体（E_1、E_2、E_5）都包含该漏洞，则这 3 个执行体受到攻击后会产生 3 个一致的异常输出。由于其数量大于系统表决模 2，因而攻击者可以成功绕过表决器，完成攻击。

从以上案例中可看出，若服务体 S 中有 N_v 个执行体存在共同漏洞 v，则当攻击者针对漏洞 v 发起攻击时，存在以下 2 种情形。

1) 当 $N_v < k$ 时，表决器有效阻断该 N_v 个一致的异常输出，从而成功抵御了针对该共同漏洞 v 的攻击。

2) 当 $N_v \geqslant k$ 时，表决器将 N_v 个一致的异常输出判定为正常输出，攻击能有效绕过表决器，从而攻击成功。

产生以上问题的原因[9]在于 DHR 中动态选择算法对执行体的选择是随机的，并不考虑执行体之间的具体漏洞分布，从而无法得到包含较少共同漏洞的服务体。

对于以上由已知共同漏洞所导致的 DHR 系统脆弱性问题，虽然可以通过漏洞修补得到解决，但是对于一个信息系统来说，存在未知漏洞是一个必然现象。由于目前仍然无法通过测试发现系统中存在的所有漏洞[22-23]，因此由共同漏洞引起的系统脆弱性问题必然存在。

4　IDHR 架构及安全性分析

4.1　IDHR 架构

为了应对因共同漏洞导致的 DHR 系统脆弱性问题，本文在现有架构基础上，提出一种改进的 DHR 架构——IDHR，如图 2 所示。具体改进如下。

1) 在 DHR 架构的调度模块和构建模块之间引入执行体划分模块。该模块根据执行体的异构性，将执行体集 E 中的执行体划分为多个执行体子池，使同一子池内的执行体间的共同漏洞尽量多，且不同子池中的执行体之间共同漏洞尽量少。

2) 改进了调度模块中的动态选择算法。系统运行时，调度模块先从多个异构执行体子池中随机选择 n 个执行体子池，然后分别从被选定的 n 个执行体子池中随机选择一个执行体，以此构成 IDHR 处理模块中的服务体。

上述改进的核心是找到执行体所包含漏洞的表示模型以及执行体之间异构性的度量指标。对此，本节参考了软硬件聚类领域中常用的二进制序列表示方法[24-27]，给出执行体所包含漏洞的表示模型——执行体–漏洞向量。

定义 1　执行体–漏洞向量。记 DHR 系统包含的所有漏洞为 $V=\{v_1,v_2,\cdots,v_\omega\}$，定义执行体 E_i 的漏洞向量为

$$\boldsymbol{\alpha}_i=[\mathrm{qv}_{i1},\cdots,\mathrm{qv}_{ij},\cdots,\mathrm{qv}_{i\omega}],1\leqslant j\leqslant\omega$$

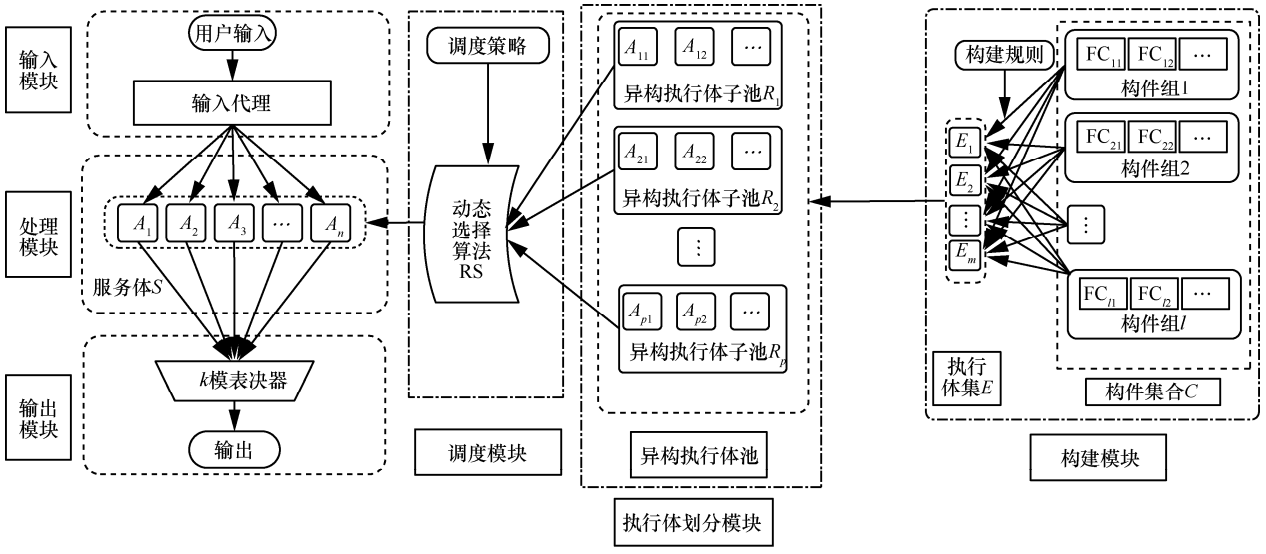

图 2　IDHR 架构

其中，E_i 存在漏洞 v_j 时，$\mathrm{qv}_{ij}=1$；否则 $\mathrm{qv}_{ij}=0$。

基于执行体−漏洞向量距离的执行体异构性度量指标，其本质上是一种二进制向量的相似度量。目前主要的二进制向量相似度量指标有欧几里得距离、汉明距离、余弦相似度和 Jaccard 距离[28-29]。其中，最常用的 Jaccard 距离定义为

$$\mathrm{JacDist}(\boldsymbol{\alpha}_i,\boldsymbol{\alpha}_j)=1-\frac{\left\|\boldsymbol{\alpha}_i+\boldsymbol{\alpha}_j\right\|}{\left|\boldsymbol{\alpha}_i+\boldsymbol{\alpha}_j\right|} \tag{1}$$

其中，$\left\|\boldsymbol{\alpha}_i+\boldsymbol{\alpha}_j\right\|$ 表示向量 $\boldsymbol{\alpha}_i+\boldsymbol{\alpha}_j$ 各分量的值等于 2 的分量数，$\left|\boldsymbol{\alpha}_i+\boldsymbol{\alpha}_j\right|$ 表示向量 $\boldsymbol{\alpha}_i+\boldsymbol{\alpha}_j$ 各分量的值大于 0 的分量数。

距离计算方法影响执行体异构性度量的准确性。当 2 个执行体 E_i 和 E_j 对应的漏洞向量 $\boldsymbol{\alpha}_i$ 和 $\boldsymbol{\alpha}_j$ 在某一分量上的值相同，即两者都存在或都不存在某个共同漏洞时，执行体异构性度量有以下 2 种情况。

1) $\boldsymbol{\alpha}_i$ 和 $\boldsymbol{\alpha}_j$ 在该分量上的值均为 1，即 E_i 和 E_j 均存在漏洞 v，显然此时两者异构性较弱。

2) $\boldsymbol{\alpha}_i$ 和 $\boldsymbol{\alpha}_j$ 在该分量上的值均为 0，即当两者负匹配时[30]，E_i 和 E_j 均不存在漏洞 v，显然此时难以判断 E_i 和 E_j 的异构性。

要使划分后同一子池内存在的共同漏洞尽量多，不同子池间存在的共同漏洞尽量少，应着重考虑漏洞向量非负匹配情况。Sneath 等[30]指出，由于不考虑负匹配情况，Jaccard 距离与其他距离相比能更准确地度量软件二进制向量的差异。为此，本文基于 Jaccard 距离给出执行体−漏洞向量间的漏洞距离定义。

定义 2 执行体−漏洞向量间的距离。设执行体 E_i 和 E_j 的漏洞向量分别为 $\boldsymbol{\alpha}_i=[\mathrm{qv}_{i1},\mathrm{qv}_{i2},\cdots,\mathrm{qv}_{i\omega}]^{\mathrm{T}}$ 和 $\boldsymbol{\alpha}_j=[\mathrm{qv}_{j1},\mathrm{qv}_{j2},\cdots,\mathrm{qv}_{j\omega}]^{\mathrm{T}}$，定义 $\boldsymbol{\alpha}_i$ 和 $\boldsymbol{\alpha}_j$ 的距离为

$$\mathrm{VulDist}(\boldsymbol{\alpha}_i,\boldsymbol{\alpha}_j)=\mathrm{JacDist}(\boldsymbol{\alpha}_i,\boldsymbol{\alpha}_j)$$

根据执行体间漏洞向量的距离，将执行体集划分为多个执行体子池。算法采用 Ferdous 等[31]提出的聚类初始化方法，选择与其余执行体−漏洞向量距离最大的执行体作为子池的初始中心；利用 k-medoids 算法[32]的聚类中心更新方法，以与子池内其余执行体−漏洞向量距离最小的执行体作为新的子池中心，解决传统 k-means 算法无法在 Jaccard 距离度量下得到聚类中心的问题。

算法 1 执行体子池划分（ESP, executor sub-pool partition）算法

输入 执行体集 $E=\{E_1,E_2,\cdots,E_m\}$，最大执行体子池数 p_{thresh}（p_{thresh} 需根据要构建的拟态系统实际情况进行选取，如拟态 Web 系统一般建议设置为 10）

输出 执行体子池个数 p，执行体子池集 R，子池中的执行体到对应子池中心的距离之和与各执行体池中心间的距离之和的比值 I

1) $R\leftarrow\phi, I_{\max}\leftarrow 0, p_{\max}\leftarrow 0$

2) for p in $\{3,4,\cdots,p_{\mathrm{thresh}}\}$

3) 选择 p 个与其余执行体漏洞向量距离最大的执行体作为执行体子池的初始中心 $c_i, i=\{1,2,\cdots,p\}$

4) do

5) for j in $\{1,2,\cdots,m\}$

6) 计算执行体 E_j 和各子池中心 c_i 之间的距离 $\mathrm{VulDist}(E_j,c_i)$

7) 设 c_i 为距离 E_j 最近的中心，其所对应的执行体子池为 R_i，将 E_j 指派到 R_i：$R_i=R_i\cup\{E_j\}$

8) end for

9) 更新子池中心。将各子池内与其余执行体距离之和最小的执行体作为该子池中心

$$c_i=\underset{E_a\in R_i}{\operatorname{argmin}}\sum_{E_b\in R_i}\mathrm{VulDist}(E_a,E_b)$$

10) until 子池中心未发生变化或达到最大迭代次数

11) 计算各子池中的执行体到对应子池中心的距离的平均值 IntraDist，以及各执行体池中心间的距离的平均值 InterDist

12) $I\leftarrow \mathrm{InterDist/IntraDist}$

13) if $I>I_{\max}$

14) $p_{\max}\leftarrow p$

15) $R\leftarrow\{R_1,R_2,\cdots,R_{p_{\max}}\}$

16) $I_{\max}\leftarrow I$

17) end if

18) end for

由以上算法描述可知，ESP 算法是从已知漏洞出发来刻画执行体的异构性，进而达到将执行体集划分为多个具有较大异构性的执行体子池。但如前所述，信息系统存在未知漏洞是一个必然现象。接

下来，通过已知漏洞和未知漏洞之间关联性实验验证分析来说明 ESP 算法对未知漏洞的有效性。

验证实验在 CVE 漏洞库中选择 2010—2018 年内与操作系统、数据库软件、服务器软件和后台脚本语言相关的漏洞作为已知漏洞，从 2019 年 1 月—2020 年 4 月的漏洞中随机选择 100 个漏洞作为“未知”漏洞，利用 5.1.2 节中所述的方法构建执行体集 E，然后用 ESP 算法进行执行子池的划分，结果如表 1 所示。

当未考虑“未知”漏洞时，ESP 算法所获得的执行体子池 1 中 43 个执行体；当考虑“未知”漏洞时，仍有 39 个执行体在子池 1 中，占比为 $39/43=0.91$。同样可得，其余 3 个子池中，当包含“未知”漏洞时，执行体仍在对应子池中的比例分别为 1、0.87 和 1。由此可得，当包含已知及“未知”漏洞时，相较于仅包含已知漏洞情形，大约有 94%的执行体仍在对应子池内。

由上述结果可知，未知漏洞的分布与已知漏洞分布之间存在非常大的关联性，这也说明利用已知漏洞对执行体池进行划分是合理的。

4.2　IDHR 安全性分析

本节参考文献[33]提出的概率分析方法，对 IDHR 安全性进行分析。

4.2.1　相关定义和假设

本文的安全性分析基于黑盒模型[33]，即攻击者不了解系统的内在结构。该模型下，具体假设如下。

1) 一次攻击仅利用一个漏洞。

2) 若系统存在某一漏洞，且攻击者针对该漏洞进行攻击，则攻击必定成功。

3) 系统选择所有服务体的概率相同。

4) 攻击采用盲攻击（即等概率选择攻击方式）策略。

接下来，给出系统的安全性度量指标、执行体−漏洞矩阵和服务体−漏洞矩阵等相关定义，其中，系统的安全性度量指标具体包括攻击成功率和系统被控制率。

定义 3　系统攻击成功率[10]。在系统调度周期 t_s 内，若攻击者至少攻击成功一次，称该周期内系统被成功攻击。给定攻击者总攻击次数 δ，其中攻击成功的次数为 λ，则称 $\mathrm{SPr}=\lambda/\delta$ 为系统攻击成功率。

定义 4　系统被控制率[10]。在单个系统调度周期 t_s 内，攻击者从开始攻击至其第一次攻击成功所耗费的时间为 Δ_t，称 $t_s-\Delta_t$ 为系统被控制时间。系统被控制时间与运行时间之比被称为系统被控制率，记为 $\theta=(t_s-\Delta_t)/t_s$。

定义 5　执行体−漏洞矩阵。对于执行体集 $E=\{E_1,\cdots,E_m\}$，称

$$\mathbf{EV}_{m\times\omega}=\left[\boldsymbol{\alpha}_1\quad\boldsymbol{\alpha}_2\quad\cdots\quad\boldsymbol{\alpha}_m\right]^{\mathrm{T}}$$

为对应的执行体−漏洞矩阵，其中，$\boldsymbol{\alpha}_1,\boldsymbol{\alpha}_2,\cdots,\boldsymbol{\alpha}_m$ 为执行体 $E=\{E_1,\cdots,E_m\}$ 的漏洞向量。

定义 6　服务体−漏洞矩阵。对于服务体集 $S=\{S_1,S_2,\cdots,S_l\}$，称

$$\mathbf{SV}_{l\times\omega}=\left[\boldsymbol{\beta}_1\quad\boldsymbol{\beta}_2\quad\cdots\quad\boldsymbol{\beta}_l\right]^{\mathrm{T}}$$

为对应的服务体−漏洞矩阵，其中，$\boldsymbol{\beta}_1,\boldsymbol{\beta}_2,\cdots,\boldsymbol{\beta}_l$ 为服务体 $\{S_1,S_2,\cdots,S_l\}$ 的漏洞向量。

接下来，构建例 1 所示系统的 IDHR 架构中的 $\mathbf{EV}_{m\times\omega}$ 和 $\mathbf{SV}_{l\times\omega}$，如例 2 所示。

例 2　$\mathbf{EV}_{m\times\omega}$ 和 $\mathbf{SV}_{l\times\omega}$ 的构建。

由定义 3 可得，执行体−漏洞矩阵为

$$\mathbf{EV}_{m\times\omega}=\left[\boldsymbol{\alpha}_1\quad\boldsymbol{\alpha}_2\quad\boldsymbol{\alpha}_3\quad\boldsymbol{\alpha}_4\quad\boldsymbol{\alpha}_5\quad\boldsymbol{\alpha}_6\right]^{\mathrm{T}}=\begin{bmatrix}1&0&0&1&1\\1&1&0&0&0\\0&0&1&1&0\\0&1&0&0&1\\1&0&1&0&1\\0&1&0&1&0\end{bmatrix}$$

利用 ESP 算法得到的各执行体子池为

表 1　2 种情形下子池划分的混淆矩阵

包含已知漏洞时，各执行体子池划分情况	包含“未知”漏洞时，各执行体子池划分情况			
	子池 1	子池 2	子池 3	子池 4
子池 1	39	4	0	0
子池 2	0	25	0	0
子池 3	2	2	29	0
子池 4	0	0	0	16

$R=\{R_1,R_2,R_3,R_4\}=\{\{E_1\},\{E_2,E_3,E_6\},\{E_4\},\{E_5\}\}$

按字典序从 R 中随机选择 3 个子池的组合有 $\{R_1,R_2,R_3\}$、$\{R_1,R_2,R_4\}$、$\{R_2,R_3,R_4\}$ 和 $\{R_1,R_3,R_4\}$。在上述组合中，从每个池中按字典序依次选择执行体，最终得到系统的服务体集 S 为

$$\begin{aligned}S=&\{S_1,S_2,\cdots,S_{10}\}=\\&\{\{E_1,E_2,E_4\},\{E_1,E_3,E_4\},\{E_1,E_4,E_6\},\\&\{E_1,E_2,E_5\},\{E_1,E_3,E_5\},\{E_1,E_5,E_6\},\\&\{E_2,E_4,E_5\},\{E_3,E_4,E_5\},\{E_4,E_5,E_6\},\\&\{E_1,E_4,E_5\}\}\end{aligned}$$

以服务体 $S_1=\{E_1,E_2,E_4\}$ 为例，其对应的执行体−漏洞矩阵为

$$\left(\mathbf{EV}_{n\times\omega}\right)_{S_1}=\begin{bmatrix}\boldsymbol{\alpha}_1^{\mathrm{T}}\\ \boldsymbol{\alpha}_2^{\mathrm{T}}\\ \boldsymbol{\alpha}_4^{\mathrm{T}}\end{bmatrix}=\begin{bmatrix}1&0&0&1&1\\1&1&0&0&0\\0&1&0&0&1\end{bmatrix}$$

因为表决模 $k=2$，所以服务体 S_1 的共同漏洞为 $\{v_1,v_2,v_5\}$，即 S_1 的漏洞向量为

$$\boldsymbol{\beta}_1=\begin{bmatrix}1&1&0&0&1\end{bmatrix}^{\mathrm{T}}$$

最终的服务体−漏洞矩阵为

$$\mathbf{SV}_{l\times\omega}=[\boldsymbol{\beta}_1\quad\boldsymbol{\beta}_2\quad\boldsymbol{\beta}_3\quad\cdots\quad\boldsymbol{\beta}_{10}]^{\mathrm{T}}=\begin{bmatrix}1&0&0&1&1&1&1&0&0&1\\1&0&1&0&0&0&1&0&1&0\\0&0&0&0&1&0&0&1&0&0\\0&1&1&0&1&1&0&0&0&0\\1&1&1&1&1&1&1&1&1&1\end{bmatrix}^{\mathrm{T}}$$

4.2.2 IDHR 系统攻击成功率

接下来，对合谋攻击策略下 IDHR 系统攻击成功率和被控制率的具体计算进行分析。

首先，分析单攻击者攻击 τ 次时的系统攻击成功率。如文献[33]中所述，虽然可以通过枚举攻击者的所有攻击方式计算系统攻击成功率和被控制率，但其计算复杂度为 $O(P_\omega^\tau)$，其中 P 表示排列数。为降低计算复杂度，本文提出先分别计算各个服务体的 τ 次攻击成功率，然后将其进行加权平均，最终得到系统攻击成功率和被控制率。

基于各服务体系统攻击成功率加权平均的系统攻击成功率 SPr 计算式为

$$\mathrm{SPr}=\sum_{S_i\in S}\Pr(S_i)\mathrm{SPr}_\tau(S_i)\tag{2}$$

其中，$\Pr(S_i)$ 表示每个服务体被调度的概率，$\mathrm{SPr}_\tau(S_i)$ 表示攻击者对 S_i 进行 τ 次攻击至少一次成功的概率。

当单个攻击者攻击 τ 次时，因为系统总漏洞数为 ω，记服务体 S_i 中存在的漏洞数为 η_i，则对 S_i 进行 τ 次攻击均失败的情况有 $P_{\omega-\eta_i}^{\min(\omega,\tau)}$ 种（即攻击者在服务体为 S_i 时选择了 $\min(\omega,\tau)$ 个漏洞进行攻击，且攻击失败），则对 S_i 进行 τ 次攻击均失败的概率 $\mathrm{FPr}_\tau(S_i)$ 和至少成功一次的概率 $\mathrm{SPr}_\tau(S_i)$ 分别为

$$\mathrm{FPr}_\tau(S_i)=\frac{P_{\omega-\eta_i}^{\min(\omega,\tau)}}{P_\omega^{\min(\omega,\tau)}}\tag{3}$$

$$\mathrm{SPr}_\tau(S_i)=1-\mathrm{FPr}_\tau(S_i)=1-\frac{P_{\omega-\eta_i}^{\min(\omega,\tau)}}{P_\omega^{\min(\omega,\tau)}}\tag{4}$$

当执行体子池个数为 p 时，IDHR 调度模块所有子池中随机选择 n 个执行体池的方法共有 C_p^n 种。记 $R_{uy}(1\leqslant y\leqslant n)$ 为由第 $u(1\leqslant u\leqslant \mathrm{C}_p^n)$ 种选择方法所确定的子池，R_{uy} 中执行体的个数记为 r_{uy}，可得该选择方法下 n 个执行体子池中执行体的总数 l_u 为

$$l_u=\prod_{y=1}^{n}r_{uy}\tag{5}$$

因为 IDHR 调度模块等概率地选择各服务体，由式(5)可得

$$\Pr(S_i)=\frac{1}{\sum\limits_{u=1}^{\mathrm{C}_p^n}\prod\limits_{y=1}^{n}r_{uy}}\tag{6}$$

合谋盲攻击策略下，由于多个攻击者共享攻击方式，不会对同一个漏洞进行多次攻击，因此该策略下，可将多个攻击者对系统发起 τ 次攻击视为单个攻击者对系统发起 $N\lceil t_s/t_a\rceil$ 次攻击。由于执行体最多包含 ω 种漏洞，则攻击者在调度时间内的最大有效攻击次数为

$$\tau_{\max}=\min\left(\omega,N\left\lceil\frac{t_{\mathrm{s}}}{t_{\mathrm{a}}}\right\rceil\right)\tag{7}$$

由式(2)～式(7)可得，合谋盲攻击策略下的攻击成功率为

$$\mathrm{SPr}=\frac{\sum\limits_{S_i\in S}\left(1-\dfrac{P_{\omega-\eta_i}^{\tau_{\max}}}{P_\omega^{\tau_{\max}}}\right)}{\sum\limits_{u=1}^{\mathrm{C}_p^n}\prod\limits_{y=1}^{n}r_{uy}}\tag{8}$$

4.2.3 IDHR 系统被控制率

根据定义 6，可得单个攻击者攻击 τ 次时的系统被控制率为

$$\theta_\tau = \frac{t_s - \tau t_a}{t_s}\mathrm{OSPR}_\tau \tag{9}$$

其中，OSPR_τ 是指攻击者在第 $1\sim\tau-1$ 次攻击均失败且仅第 τ 次攻击成功的概率。与 4.2.2 节提出的攻击成功率计算方法类似，OSPR_τ 也可通过式(10)计算。

$$\mathrm{OSPr}_\tau = \sum_{S_i\in S}\Pr(S_i)\mathrm{OSPr}_\tau(S_i) \tag{10}$$

其中，$\mathrm{OSPr}_\tau(S_i)$ 表示服务体 S_i 在攻击者的 τ 轮攻击中仅第 τ 轮被攻破的概率。

合谋盲攻击策略下，当 N 个攻击者进行 τ 轮攻击时，其在 $\tau-1$ 轮合谋攻击所针对的漏洞数为 $N(\tau-1)$。针对服务体 S_i 的前 $\tau-1$ 轮攻击失败的可能数为 $P_{\omega-\eta_i}^{N(\tau-1)}$，而第 τ 轮攻击获得成功的次数可能有 $1\sim\min(N,\omega-N(\tau-1))$ 种。记攻击成功次数为 j，其在 N 次中所处位置的可能有 C_N^j 种，每种位置对应的攻击方式有 $P_{\eta_i}^j P_{\omega-\eta_i-N(\tau-1)}^{\min(iN,\omega)-j}$ 种，因此攻击成功 j 次对应攻击序列总共有 $\mathrm{C}_N^j P_{\eta_i}^j P_{\omega-\eta_i-N(\tau-1)}^{\min(iN,\omega)-j}$ 种。由于在盲攻击下，各个漏洞被选取的概率相等，因此服务体 S_i 在 N 个攻击者的 τ 次攻击中，仅第 τ 次被攻破的概率 $\mathrm{OSPr}_\tau(\mathrm{S}_i)$ 为

$$\mathrm{OSPr}_\tau(S_i) = \frac{\sum_{j=1}^{\min(N,\omega-N(\tau-1))}\left(\mathrm{C}_N^j P_{\eta_i}^j P_{\omega-\eta_i}^{\min(iN,\omega)-j}\right)}{P_\omega^{\min(\omega,N\tau)}} \tag{11}$$

由于合谋盲攻击策略下攻击者针对一个 IDHR 服务体的最大有效攻击次数为 $\tau_{\max}=\min\left(\left\lceil\frac{\omega}{N}\right\rceil,\left\lceil\frac{t_s}{t_a}\right\rceil\right)$，因此系统被控制率计算式为

$$\theta=\sum_{q=1}^{\tau_{\max}}\theta_q=\sum_{q=1}^{\min\left(\left\lceil\frac{\omega}{N}\right\rceil,\left\lceil\frac{t_s}{t_a}\right\rceil\right)}\frac{t_s-qt_a}{t_s}\cdot \sum_{S_i\in S}\frac{\sum_{j=1}^{\min(N,\omega-N(q-1))}\left(\mathrm{C}_N^j P_{\eta_u}^j P_{\omega-\eta_u}^{\min(qN,\omega)-j}\right)}{P_\omega^{\min(\omega,Nq)}\sum_{u=1}^{\mathrm{C}_p^n}\prod_{y=1}^{n}r_{uy}} \tag{12}$$

5　实验和分析

本节通过实验对 IDHR 系统安全性进行量化评估，并与 DHR 系统安全性进行比较分析。实验包括随机模拟执行体和仿真 Web 服务器 2 种方案，每种方案均包含存在和不存在未知漏洞 2 种情况。本节首先给出总体实验方案设计，并根据总体方案，给出随机模拟执行体和仿真 Web 服务器 2 种具体方案的设计。然后分别得到 2 种方案下，DHR 和 IDHR 系统存在和不存在未知漏洞时的攻击成功率和被控制率，并进行比较。最后对实验结果进行分析总结。

5.1　实验总体方案

由于实际中未知漏洞无法被探知，因此本文在随机模拟执行体和仿真 Web 服务器 2 种方案中，通过对应方法（具体见 5.1.1 节和 5.1.2 节），以模拟系统存在未知漏洞的情形。

实验总体方案的步骤如下。

1) 生成已知漏洞的执行体−漏洞矩阵 $\mathbf{EV}_{n\times\omega}$。此时 DHR 和 IDHR 这 2 种架构的执行体−漏洞矩阵相同。

2) 根据 $\mathbf{EV}_{n\times\omega}$，构建已知漏洞的 DHR 和 IDHR 的服务体−漏洞矩阵 $\mathbf{SV}_{l\times\omega}^{(\mathrm{DHR})}$ 和 $\mathbf{SV}_{l\times\omega}^{(\mathrm{IDHR})}$。

3) 利用 5.1.1 节和 5.1.2 节所述方法，生成“未知”漏洞所对应的执行体−漏洞向量，从而在步骤 1)所得的 $\mathbf{EV}_{n\times\omega}$ 基础上，得到包含已知及“未知”漏洞的执行体−漏洞矩阵 $\mathbf{EV}'_{n\times(\omega+\mu)}$。

4) 根据 $\mathbf{EV}'_{n\times(\omega+\mu)}$，生成系统存在“未知”漏洞时的服务体−漏洞矩阵 $\mathbf{SV}'^{(\mathrm{DHR})}_{l\times(\omega+\mu)}$、$\mathbf{SV}'^{(\mathrm{IDHR})}_{l\times(\omega+\mu)}$。

5) 根据 $\mathbf{SV}_{l\times\omega}^{(\mathrm{DHR})}$、$\mathbf{SV}_{l\times\omega}^{(\mathrm{IDHR})}$、$\mathbf{SV}'^{(\mathrm{DHR})}_{l\times(\omega+\mu)}$ 和 $\mathbf{SV}'^{(\mathrm{IDHR})}_{l\times(\omega+\mu)}$，并指定相关参数 $\{t_s,t_a,N\}$，根据式(8)、式(12)计算 IDHR 和 DHR 架构在合谋攻击策略下的攻击成功率和被控制率。

在总体方案的步骤 2)中，IDHR 的服务体−漏洞矩阵 $\mathbf{SV}_{l\times\omega}^{(\mathrm{IDHR})}$ 的构建参考例 2。DHR 服务体−漏洞矩阵 $\mathbf{SV}_{l\times\omega}^{(\mathrm{DHR})}$ 的构建方式[33]如下。

1) 从执行体−漏洞矩阵中任取 n 行，得到 C_m^n 个子矩阵，记为 $\{\boldsymbol{M}_1,\boldsymbol{M}_2,\cdots,\boldsymbol{M}_l\}$。

2) 将每个子矩阵的所有行相加，得到 C_m^n 个行向量 $\boldsymbol{m}_i$，$1\leqslant i\leqslant l$。

3) 对于每个行向量 $\boldsymbol{m}_i=\{m_{i1},\cdots,m_{i\omega}\}$，令

$$m'_{ij}=\begin{cases}1, m_{ij}\geqslant k\\ 0, m_{ij}<k\end{cases}$$

则服务体−漏洞矩阵为

$$\mathbf{SV}_{l\times\omega}^{(\mathrm{DHR})}=(m'_{ij})_{l\times\omega}$$

总体方案步骤 4)中 $\mathbf{SV}'^{(\mathrm{DHR})}_{l\times(\omega+\mu)}$ 的构建过程与总体方案的步骤 2)相同，$\mathbf{SV}'^{(\mathrm{IDHR})}_{l\times(\omega+\mu)}$ 的构建过程如下。

1) 利用总体方案的步骤 1)中得到的 $\mathbf{EV}_{n\times\omega}$，根据 ESP 算法，得到各执行体子池为 $\{R_1,R_2,\cdots,R_p\}$。

2) 根据 $\{R_1,R_2,\cdots,R_p\}$ 和总体方案的步骤 3)中得到的 $\mathbf{EV}'_{n\times(\omega+\mu)}$，参考本文例 2，生成 $\mathbf{SV}'^{(\mathrm{IDHR})}_{l\times(\omega+\mu)}$。

5.1.1 随机模拟执行体方案

本文方案首先给定如下模拟系统构建参数，如表 2 所示，然后通过随机模拟的方式生成执行体−漏洞矩阵。

表 2 模拟系统构建参数

系统	参数	含义
DHR 或 IDHR	m	执行体数
	n	选择模
	k	判决模
	p_{vul}	漏洞产生概率
	ω	已知漏洞个数
	μ	未知漏洞个数
IDHR	p	执行体子池个数

构建执行体−漏洞矩阵的具体方式如下。

1) 以漏洞产生概率 p_{vul} 的均匀分布生成 m 个 ω 维的行向量 $\boldsymbol{\alpha}_1,\boldsymbol{\alpha}_2,\cdots,\boldsymbol{\alpha}_m$，作为 m 个执行体的执行体−漏洞向量。

2) 由 $\boldsymbol{\alpha}_1,\boldsymbol{\alpha}_2,\cdots,\boldsymbol{\alpha}_m$ 组成执行体−漏洞矩阵 $\mathbf{EV}_{n\times\omega}$。

随机模拟执行体方案中系统“未知”漏洞的生成和执行体−漏洞矩阵的构建方式具体如下。

1) 以漏洞产生概率 p_{vul} 的均匀分布构建 m 个 μ 维的执行体−漏洞向量 $\boldsymbol{\alpha}'_1,\boldsymbol{\alpha}'_2,\cdots,\boldsymbol{\alpha}'_m$。

2) 将 $\boldsymbol{\alpha}'_1,\boldsymbol{\alpha}'_2,\cdots,\boldsymbol{\alpha}'_m$ 按行合并至原执行体−漏洞矩阵，得到系统存在未知漏洞时的执行体−漏洞矩阵 $\mathbf{EV}'_{n\times(\omega+\mu)}$。

本文根据实际情况设定模拟实验的参数，通过 CVE 漏洞库得到的相关参数具体为 $p_{\mathrm{vul}}=0.17$，$\omega=3\,300$，$\mu=100$，$m=117$，$n=3$，$k=2$，$p=4$，$N=20$，$t_s=20$，$t_a=1$。由于在实际 Web 构建中，会对已知漏洞进行修补，因此实际 Web 系统的漏洞出现概率 p_{vul} 和漏洞向量维数 ω 都比 Web 仿真服务器方案小一些，而随机模拟执行体方案的参数也与实际接近。实际 Web 系统的相关参数具体为 $p_{\mathrm{vul}}=0.03$，$\omega=1\,000$，$\mu=100$，$m=30$，$n=3$，$k=2$，$p=4$，$N=20$，$t_s=20$，$t_a=1$。

5.1.2 仿真 Web 服务器方案

本文方案首先根据操作系统、服务器软件、后台脚本语言和数据库软件 4 个类型，从通用平台枚举（CPE, common platform enumeration）数据库中选择对应的软件，构建构件集。CPE 数据库由美国国家标准与技术研究院（NIST, National Institute of Standards and Technology）托管和维护，是一个记录信息系统中各种软件的数据库。然后根据 Web 服务器软件栈[12-13]，从构件集中选择相应构件组成执行体集。

截至 2020 年 4 月，主流 Web 服务器相关软件包发行版（不包括测试版）的版本数目为操作系统 752 个、服务器软件 719 个、后台脚本语言 2 132 个和数据库软件 1 376 个（如表 3 所示）。但为了使服务更加稳定，人们并非随机选择 Web 服务器相关软件的版本，而是优先选择业界认可的相关软件版本，如 Nginx1.15.8。此外，由于相近版本软件包含的漏洞通常大致相同，同时若选取全部软件来构建拟态 Web 服务器模型会耗费大量时间。因此，本文方案按软件的漏洞相似性（使用 Jaccard 距离进行度量）对各软件版本进行聚类，同时结合软件使用率对软件进行筛选。由于篇幅所限，本节仅展示部分 Web 服务器相关软件的聚类结果，分别如图 3 和图 4 所示。本文得到的全部 Web 服务器相关软件发行版按漏洞相似性聚类的数据结果见 GitHub 开源仓库 idhrcve-data。

在软件版本选择后，本文最终选择了 6 种操作系统、4 种服务器软件、3 种后台脚本语言和 5 种数据库软件（如表 4 所示），构建了 117 个 Web 服务器执行体。

仿真 Web 服务器方案的执行体−漏洞矩阵生成方式如下。

1) 从 CVE 漏洞库获取 2010—2018 年所有执行体中软件对应的 CVE 漏洞，构成执行体漏洞集。

2) 根据定义 1 生成所有执行体−漏洞向量，将这些执行体−漏洞向量组成执行体−漏洞矩阵。

仿真 Web 服务器方案的“未知”漏洞的生成和执行体−漏洞矩阵的构建方式如下。

1) 从 2019 年 1 月—2020 年 4 月所有执行体中软件对应的 CVE 漏洞中等概率随机选择 100 个漏洞，作为系统未知漏洞，生成所有执行体−漏洞向量 $\boldsymbol{\alpha}'_1,\boldsymbol{\alpha}'_2,\cdots,\boldsymbol{\alpha}'_m$。

表 3　常用 Web 服务器相关软件包发行版本数

软件类型	软件名称	版本数/个
操作系统	Windows Server	424
	Ubuntu	93
	Suse Linux	34
	RadHat Enteprise Linux	144
	Debian	33
	CentOS	24
后台脚本语言	Python	355
	PHP	893
	Java(jre)	256
	Node.js	628
数据库软件	MySQL	670
	MariaDB	182
	PostgreSQL	409
	Oracle Database Server	115
服务器软件	Nginx	511
	Apache	195
	IIS	13

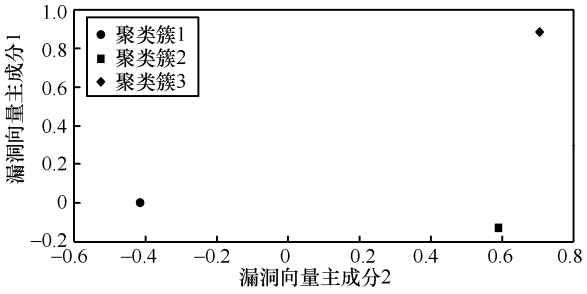

图 3　CentOS 各发行版按漏洞相似性聚类结果

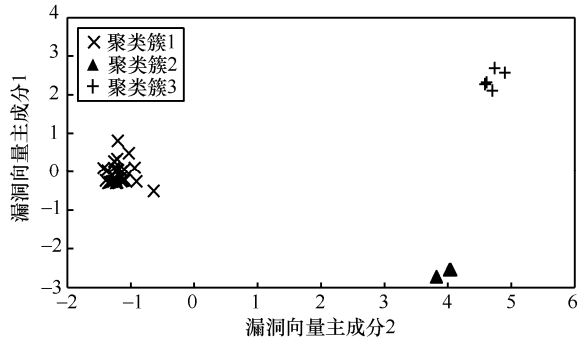

图 4　SQL Server 各发行版按漏洞相似性聚类结果

2) 按行合并至原执行体−漏洞矩阵 $\mathbf{EV}_{n\times\omega}$，得到系统存在未知漏洞时的执行体−漏洞矩阵 $\mathbf{EV}'_{n\times(\omega+\mu)}$。

5.2　实验结果

图 5 和图 6 分别为随机模拟执行体和仿真 Web 服务器 2 种方案的实验结果。

表 4　IDHR 架构的 Web 服务器系统软件选择

软件类型	软件名称	具体版本
操作系统	Windows Server	windows_server_2019 windows_server_2016:1803
	Suse Linux	suse_linux:10.0
	RedHat Enterprise Linux	enterprise_linux:8.0
	Debian	debian_linux:7.0 ian_linux:10
后台脚本语言	Python	python:3.5.0
	PHP	php:7.3.1
	Java	jre:8.0
服务器软件	Nginx	nginx:1.6.1 nginx:1.13.8
	Apache	http_server:2.2.17
	IIS	internet_information_server:10.0
数据库软件	MySQL	mysql:5.3.4 sql:8.0.1
	Oracle Database Server	database_server:9.0.1.5 database_server:12.1.0.1
	SQL Server	sql_server:2019

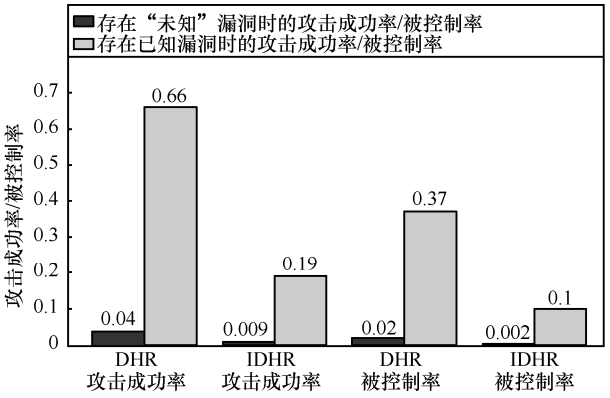

图 5　随机模拟执行体方案下，IDHR 和 DHR 在合谋盲攻击下的攻击成功率和被控制率结果对比

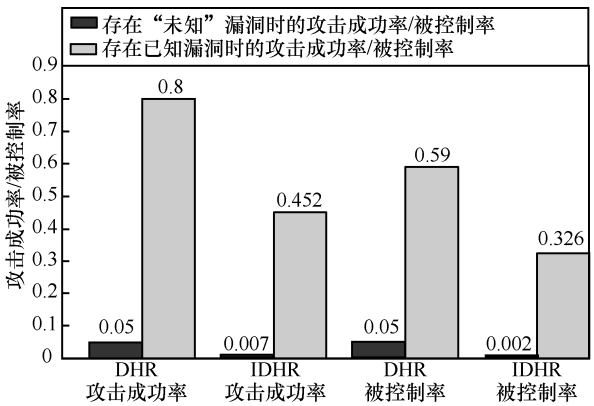

图 6　仿真 Web 服务器方案下，IDHR 和 DHR 在合谋盲攻击下的攻击成功率和被控制率结果对比

如图 5 所示，在存在已知漏洞情况下，DHR 架构的攻击成功率和被控制率分别为 0.66 和 0.37；IDHR 架构的攻击成功率和被控制率分别为 0.19 和 0.1，相较 DHR 架构降低了约 2/3。在存在“未知”漏洞情况下，DHR 架构的攻击成功率和被控制率分别为 0.7 和 0.39；IDHR 架构的攻击成功率和被控制率分别为 0.199 和 0.102，相较 DHR 架构仍降低了约 2/3。此时，相较于存在已知漏洞情形，IDHR 的攻击成功率和被控制率分别仅增加 0.009 和 0.002，而 DHR 的攻击成功率和被控制率分别增加 0.04 和 0.02。

由图 6 所示，在存在已知漏洞情况下，DHR 架构的攻击成功率和被控制率分别为 0.8 和 0.59；IDHR 架构的攻击成功率和被控制率分别为 0.452 和 0.326，相较 DHR 架构降低了约1/2。在存在“未知”漏洞情况下，DHR 架构的攻击成功率和被控制率分别为 0.85 和 0.64；IDHR 架构的攻击成功率和被控制率分别为 0.459 和 0.328，相较 DHR 架构仍降低了约1/2，此时，相较于存在已知漏洞情形，IDHR 的攻击成功率和被控制率分别仅增加 0.007 和 0.002，而 DHR 的攻击成功率和被控制率均增加 0.05。

由上述结果可知，在随机模拟执行体和仿真 Web 服务器方案下，IDHR 相较于 DHR，在攻击成功率和被控制率上具有明显的安全优势，且存在“未知”漏洞时，IDHR 的攻击成功率和被控制率基本不变。接下来，对产生此结果的原因进行进一步分析。为此，本文对 2 种实验方案下 DHR 和 IDHR 系统的服务体包含漏洞数的分布进行了统计。

如图 7 和图 8 所示，无论是否存在“未知”漏洞，在服务体包含漏洞数的各个分布区间内，IDHR 架构中包含对应漏洞的服务体数均明显低于 DHR 架构，说明 IDHR 系统能有效减少系统中包含的共同漏洞数，因此 IDHR 架构的攻击成功率和被控制率均明显低于 DHR 架构。

此外，从图 7 和图 8 中还可以看出，存在“未知”漏洞时，IDHR 中服务体包含漏洞数的分布与存在已知漏洞时相比仍变化较小，这说明 IDHR 架构的稳健性较高。

从随机模拟执行体和仿真 Web 服务器 2 种方案的实验结果可看出，IDHR 架构的安全性明显优于 DHR 架构，尤其是存在未知漏洞时。

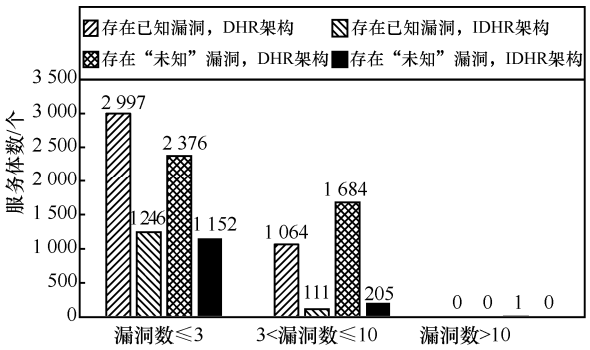

图 7 随机模拟执行体方案下，DHR 和 IDHR 系统的服务体包含漏洞数的分布

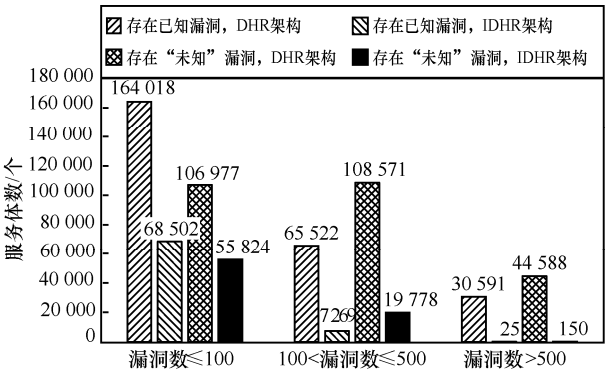

图 8 仿真 Web 服务器方案下，DHR 和 IDHR 系统的服务体包含漏洞数的分布

6 结束语

针对 DHR 系统服务体在面临共同漏洞，特别是共同漏洞未知时的系统脆弱性问题，本文提出了 IDHR 架构，引入根据执行体间的异构性对执行体集进行划分的执行体划分模块，并改进调度模块中的动态选择算法，进一步增强系统安全性。后续将基于本文所做工作，继续优化执行体划分算法，进一步提高系统安全性。

参考文献：

[1] ALBERTS C J, DOROFEE A J, CREEL R, et al. A systemic approach for assessing software supply-chain risk[C]//2011 44th Hawaii International Conference on System Sciences. Piscataway: IEEE Press, 2011: 1-8.

[2] 陈福才, 何威振, 程国振, 等. 基于 DPDK 的内网动态网关关键技术设计[J]. 通信学报, 2020, 41(6): 139-151.
CHEN F C, HE W Z, CHENG G Z, et al. Design of key technologies for intranet dynamic gateway based on DPDK[J]. Journal on Communications, 2020, 41(6): 139-151.

[3] HOUSE W. Trustworthy cyberspace: strategic plan for the federal cyber security research and development program[R]. Report of the National Science and Technology Council, Executive Office of the

President, 2011.

[4] 谭晶磊, 张恒巍, 张红旗, 等. 基于Markov时间博弈的移动目标防御最优策略选取方法[J]. 通信学报, 2020, 41(1): 42-52.
TAN J L, ZHANG H W, ZHANG H Q, et al. Optimal strategy selection approach of moving target defense based on Markov time game[J]. Journal on Communications, 2020, 41(1): 42-52.

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

[7] 邬江兴. 拟态计算与拟态安全防御的原意和愿景[J]. 电信科学, 2014, 30(7): 2-7.
WU J X. Meaning and vision of mimic computing and mimic security defense[J]. Telecommunications Science, 2014, 30(7): 2-7.

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

[9] 张杰鑫, 庞建民, 张铮, 等. 基于非相似余度架构的网络空间安全系统异构性量化方法[J]. 电子与信息学报, 2019, 41(7): 1594-1600.
ZHANG J X, PANG J M, ZHANG Z, et al. Heterogeneity quantization method of cyberspace security system based on dissimilar redundancy structure[J]. Journal of Electronics & Information Technology, 2019, 41(7): 1594-1600.

[10] 王伟, 曾俊杰, 李光松, 等. 动态异构冗余系统的安全性分析[J]. 计算机工程, 2018, 44(10): 42-45, 50.
WANG W, ZENG J J, LI G S, et al. Security analysis of dynamic heterogeneous redundant system[J]. Computer Engineering, 2018, 44(10): 42-45, 50.

[11] 扈红超, 陈福才, 王禛鹏. 拟态防御 DHR 模型若干问题探讨和性能评估[J]. 信息安全学报, 2016, 1(4): 40-51.
HU H C, CHEN F C, WANG Z P. Performance evaluations on DHR for cyberspace mimic defense[J]. Journal of Cyber Security, 2016, 1(4): 40-51.

[12] 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.

[13] 张铮, 马博林, 邬江兴. Web 服务器拟态防御原理验证系统测试与分析[J]. 信息安全学报, 2017, 2(1): 13-28.
ZHANG Z, MA B L, WU J X. The test and analysis of prototype of mimic defense in Web servers[J]. Journal of Cyber Security, 2017, 2(1): 13-28.

[14] 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.

[15] 马海龙, 伊鹏, 江逸茗, 等. 基于动态异构冗余机制的路由器拟态防御体系结构[J]. 信息安全学报, 2017, 2(1): 29-42.
MA H L, YI P, JIANG Y M, et al. Dynamic heterogeneous redundancy based router architecture with mimic defenses[J]. Journal of Cyber Security, 2017, 2(1): 29-42.

[16] 丁绍虎, 李军飞, 季新生. 基于拟态防御的 SDN 控制层安全机制研究[J]. 信息安全学报, 2019, 4(4): 84-93.
DING S H, LI J F, JI X S. Research on SDN control layer security based on mimic defense[J]. Journal of Cyber Security, 2019, 4(4): 84-93.

[17] 周清雷, 班绍桓, 韩英杰, 等. 针对物理访问控制的拟态防御认证方法[J]. 通信学报, 2020, 41(6): 80-87.
ZHOU Q L, BAN S H, HAN Y J, et al. Mimic defense authentication method for physical access control[J]. Journal on Communications, 2020, 41(6): 80-87.

[18] 任权, 贺磊, 邬江兴. 基于离散马尔可夫链的不同抗干扰系统模型分析[J]. 网络与信息安全学报, 2018, 4(4): 30-37.
REN Q, HE L, WU J X. Analysis of different anti-interference system models based on discrete time Markov chain[J]. Chinese Journal of Network and Information Security, 2018, 4(4): 30-37.

[19] 朱维军, 郭渊博, 黄伯虎. 动态异构冗余结构的拟态防御自动机模型[J]. 电子学报, 2019, 47(10): 2025-2031.
ZHU W J, GUO Y B, HUANG B H. A mimic defense automaton model of dynamic heterogeneous redundancy structures[J]. Acta Electronica Sinica, 2019, 47(10): 2025-2031.

[20] ZHANG B, CHANG X, LI J. A generalized information security model SOCMD for CMD Systems[J]. Chinese Journal of Electronics, 2020, 29(3): 417-426.

[21] 李卫超, 张铮, 王立群, 等. 基于拟态防御架构的多余度裁决建模与风险分析[J]. 信息安全学报, 2018, 3(5): 64-74.
LI W C, ZHANG Z, WANG L Q, et al. The modeling and risk assessment on redundancy adjudication of mimic defense[J]. Journal of Cyber Security, 2018, 3(5): 64-74.

[22] 中国互联网络信息中心. 第 42 次《中国互联网络发展状况统计报告》[R]. 2018.
China Internet Network Information Center. The 42-nd report of statistics on china's internet development[R]. 2018.

[23] SUBRAHMANIAN V S, OVELGONNE M, DUMITRAS T, et al. The global cyber-vulnerability report[R]. 2015.

[24] MAQBOOL O, BABRI H. Hierarchical clustering for software architecture recovery[J]. IEEE Transactions on Software Engineering, 2007, 33(11): 759-780.

[25] SHTERN M, TZERPOS V. Clustering methodologies for software engineering[J]. Advances in Software Engineering, 2012, 10: 14-32.

[26] NASEEM R, MAQBOOL O, MUHAMMAD S. An improved similarity measure for binary features in software clustering[C]//2010 Second International Conference on Computational Intelligence, Modelling and Simulation. Piscataway: IEEE Press, 2010: 111-116.

[27] NASEEM R, DERIS M M. A new binary similarity measure based on

integration of the strengths of existing measures: application to software clustering[C]//International Conference on Soft Computing and Data Mining. Berlin: Springer, 2016: 304-315.

[28] CHOI S S, CHA S H, TAPPERT C C. A survey of binary similarity and distance measures[J]. Journal of Systemics, Cybernetics and Informatics, 2010, 8(1): 43-48.

[29] JACCARD P. Étude comparative de la distribution florale dans une portion des Alpes et des Jura[J]. Bulletin De La Societe Vaudoise Des Sciences Naturelles, 1901, 37: 547-579.

[30] SNEATH P H, SOKAL R R. The principles and practice of numerical classification[M]. London: Oxford University Press, 1973.

[31] FERDOUS R. An efficient k-means algorithm integrated with Jaccard distance measure for document clustering[C]//2009 First Asian Himalayas International Conference on Internet. Piscataway: IEEE Press, 2009: 1-6.

[32] KAUFMANN L. Clustering by means of medoids[C]//International Conference on Statistical Data Analysis Based on the L1-norm and Related Methods. [S.n.:s.l.], 1987: 1-10.

[33] 郑秋华, 胡程楠, 吴铤, 等. 一种基于概率分析的拟态 DHR 模型安全性分析方法[J]. 电子学报, 2020, doi: 10.12263/DZXB.20201063.
ZHENG Q H, HU C N, WU T, et al. A security analysis approach for mimic DHR model based on probability analysis[J]. Chinese Journal of Electronics, 2020, doi: 10.12263/DZXB.20201063.

[作者简介]

吴铤（1972– ），男，浙江杭州人，博士，杭州电子科技大学教授、博士生导师，主要研究方向为拟态安全、理论密码学、工控安全。

胡程楠（1996– ），男，浙江杭州人，杭州电子科技大学硕士生，主要研究方向为拟态安全、工控安全。

陈庆南（1994– ），男，浙江宁波人，杭州电子科技大学硕士生，主要研究方向为拟态安全、工控安全。

陈安邦（1996– ），男，河南信阳人，杭州电子科技大学硕士生，主要研究方向为拟态安全、工控安全。

郑秋华（1973– ），男，浙江杭州人，博士，杭州电子科技大学讲师，主要研究方向为拟态安全理论分析、拟态 Web 服务攻防技术、工控安全。