基于拟态防御架构的多余度裁决建模与风险分析

李卫超[1], 张 铮[1], 王立群[1], 邬江兴[2]

[1]数学工程与先进计算国家重点实验室 郑州 中国 450001
[2]国家数字交换系统工程技术研究中心 郑州 中国 450002

摘要 网络空间拟态防御技术以动态异构冗余的内生安全特性作为架构核心, 通过多余度裁决方保证服务质量并阻断攻击威胁。然而, 目前对于多余度裁决的方法, 并没有对其防御的代价和风险进行有效分析和评估。本文根据拟态防御与多余度裁决模型之间的关系, 针对多余度裁决方法的防御能力、运行效率和系统恢复三方面进行建模和分析。根据模型分析方法挖掘出模型的三项指标之间的潜在关系和部分同构的部署策略下裁决模型的风险隐患问题, 并通过实验验证了该方法的有效性。最后, 根据模型的评估结果给出了实际部署意见并总结了模型的不足和改进方向。

关键词 拟态防御; 多余度裁决; 安全增益; 系统开销; 恢复能力; 风险分析
中图法分类号 TP309 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2018.09.06

The Modeling and Risk Assessment on Redundancy Adjudication of Mimic Defense

LI Weichao[1], ZHANG Zheng[1], WANG Liqun[1] WU Jiangxing[2]

[1] State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China
[2] National Digital Switching System Engineering & Technological R&D Center, Zhengzhou 450002, China

Abstract Cyber space mimic defense technology takes the endogenous security characteristics of dynamic, heterogeneous and redundancy as the core of the architecture, which uses adjudication guarantees the quality of service and blocks the attack threat. However, the current method can not effectively analyze and evaluate the cost and risk of the redundancy adjudication. To evaluate the redundancy adjudication, we first introduce the association between the mimic architecture and the redundancy adjudication. Afterwards, we analyze the defense ability, operation efficiency and system recovery of the redundancy adjudication thus finding out the model indicators' potential relationship and the deployment strategies risk. Finally, the experimental evaluation and the deficiencies of the model are discussed.

Key words Mimic defense; redundancy adjudication; security gain; overhead; resilience; risk assessment

1 背景

1.1 网络空间拟态防御

网络空间作为全球互联的基础设施, 在通信、金融、工业控制等领域的发展对世界各地的人民产生了深远的影响。然而, 在促进社会文明发展的同时, 网络空间同时存在着诸多的安全问题, 信息处理与通信系统上的漏洞设计缺陷已经成为最大的不安全因素[1]。此外, 日益频繁的恶意软件攻击以及金融和知识产权盗窃须要得到根本上的解决。传统防御手段包括防火墙、身份认证、访问控制以及入侵检测等技术, 尽管对于攻击类型、来源和手段明确的网络空间安全威胁能够进行有效的防护。但是, 对基于未知漏洞和后门攻击, 或者新型病毒木马引发的未知安全问题, 则不能形成及时有效的防护。而现有的蜜罐、沙箱等基于构造可信环境的主动防御技术, 在实际的网络空间环境中难以彻底避免未知漏洞、未知后门、未知攻击带来的安全威胁。

生物学将用色彩、纹理、外观和行为的仿真或模拟来隐匿本体特征的外在表象称为“拟态伪装”(Mimic Guise, MG)[2]。在生物界中不乏伪装高手, 海洋中的变色章鱼、丛林中的避役、碎石地中的生石

通讯作者:张铮, 博士, 副教授, Email: ponyzhang@126.com。
本课题得到国家重点研发计划网络空间安全专项(No. 2017YFB0803201), 上海市科学技术委员会科研计划项目(No. 16DZ1120502)资助。
收稿日期: 2017-11-26; 修改日期: 2018-07-23; 定稿日期: 2018-08-20

花、山地沼泽中的眼镜蛇草, 等。在生物界的拟态现象启发下, 针对目前网络空间安全缺陷存在的普遍性这一根本问题, 邬江兴教授提出了网络空间拟态防御(Cyberspace Mimic Defense, CMD)[2-5]理论, 期望能够在解决多维度的网络空间安全问题的同时, 对于潜在的网络安全设计缺陷, 在技术和威胁环境变化的情况下还可以维持系统安全的有效性, 将不可控的网络空间安全威胁问题转化为自主可控的网络空间服务鲁棒性控制问题, 从内生机制或构造层面获得对未知缺陷的主动免疫能力。

1.2 相关工作及研究现状

近年来拟态防御作为网络空间安全的新兴技术, 发展十分迅速。在文献[6]中, 叙述了拟态防御的基本理论。在文献[7]中, 对拟态防御的基本理论模型及其若干问题进行了分析和探讨。目前拟态防御的在网络空间安全的应用领域已经有了丰富的研究成果。在网络基础设施中, 研制出了包括拟态 web 服务器[8-9]、拟态防御路由器[10-11]、拟态 DNS 服务器[12], 等应用设备; 在工业控制领域提出了一种拟态安全处理机架构[13]; 在移动通信领域提出了一种虚拟化拟态防御方法[14]。针对拟态防御中的异构性构造方法, 文献[15-16]提出了利用软硬件多样性实现多源异构化处理方法; 文献[17-18]提出了利用软件多样化编译的方法对单一软件进行异构化处理方法。

然而, 目前对于拟态防御中的多余度裁决, 只对其功能和方法进行了阐述, 并没有对其防御的有效性和代价进行分析和评估。而事实上不当的引入多余度裁决模型可能会导致系统防御能力和威胁感知的效果下降。因此, 本文在先前的研究基础之上, 考虑防御能力、运行效率和系统恢复三方面, 对拟态防御多余度裁决进行建模, 分析模型变化趋势, 在此基础上指出并验证了多余度裁决的部署风险问题。

2 拟态防御与多余度裁决

2.1 多样化与随机化

现代软件设计规模越来越庞大, 其中不乏功能相同但实现方法不同的服务软件[15]。由于多样化的软件在某些方面的特异性, 特定软件出现的某些错误在其他软件行为中并不会发生, 因此使得多样化系统具备抗攻击能力[16]。在一个工作系统中组成构件的丰富程度体现了系统的多样化程度, 依据工程实现上的来源和动机的不同, 多样性大致可分为三种: 自然多样性[15-19], 自动化多样性[17-20]和可控多样性[21-22]。

将随机事件发生的不确定性导入网络空间目标对象中, 可以大幅度的提高攻击者的成本和代价, 降低攻击的可靠性[2-4]。随机化方法在系统中的构件执行过程中由指令集合、地址空间的变化引入了不确定性, 该方法在运行环境、软件和数据存在着广泛的应用。例如, 地址空间随机化技术(address space randomization, ASR)和指令集随机化技术(instruction set randomization, ISR)[23-24]能够防御攻击者挖掘软件漏洞, 其中 ASR 技术在学术和商用领域都已经趋于成熟, 应用较为广泛。

多样化与随机化方法为网络空间安全提供了新的技术手段和发展方向。应用代码随机化、软件多样化等方法, 在时间维度进行动态切换, 在空间维度上产生结构差异, 能够有效的切断攻击步骤的关联性增加攻击难度, 屏蔽攻击威胁, 为拟态防御的动态、异构、冗余构造提供了基本的技术支撑。

2.2 拟态防御架构模型

网络空间的任何安全防护措施都不能影响到目标对象的服务功能的正常提供, 但是对于目标对象的系统架构、运行机制、核心算法、存储数据、异常表现、以及可能出现的未知漏洞与后门等, 都可以通过拟态伪装的方式进行主动隐匿[2,6]。网络空间信息系统的功能可以概括为 I[P]O(Inputs-Process-Outputs)处理过程。在保证 I[P]O 处理过程不变的情况下, 即系统功能不受影响的情况下, 根据“共性设计缺陷导致共模故障属于小概率事件”, 实现非相似余度架构(Dissimilar Redundancy Architecture, DRA)[2], 如图 1 所示。

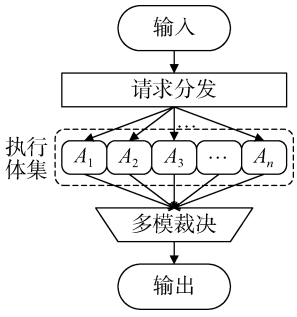

图 1 DRA 模型

Figure 1 DRA Model

DRA 利用执行构件多样化的方法, 增加执行构件间的异构性从而提高系统输出正确性。文献[25]中, 指出单一软件带来的安全风险并讨论了引入软件相异性来提高系统安全性的方法。文献[26]中, 全面分析了多种软件漏洞, 证实了大部分软件不会拥有同样的漏洞, 且即使软件拥有相同漏洞, 同一个

攻击代码仍然难以攻破该软件的不同版本。因此, 理论上如果两个构件之间的差异性足够大, 则可以保证任意一种独立的攻击方法对于不同的两个构件是不可能同时生效的。然而, 由于 DRA 对于相异性设计的要求极为严格, 且系统的运作机制仍是静态、确定性的, 所以 DRA 在应对增量性、持续性、协同性和不确定性的高阶威胁时仍面临着诸多问题[2,6]。例如, 攻击者通过利用异构程度不完全产生的共有漏洞进行协同攻击或者通过多次持续的对异构执行构件进行植入攻击都可能会使 DRA 产生安全风险。

动态异构冗余架构(Dynamic Heterogeneous Redundant Architecture, DHRA)[2]在 DRA 模型的基础上引入随机化方法, 应用动态变迁技术[27-28]实现了动态防御效果[29-30], 能够有效的解决 DRA 的风险问题, 如图 2 所示。

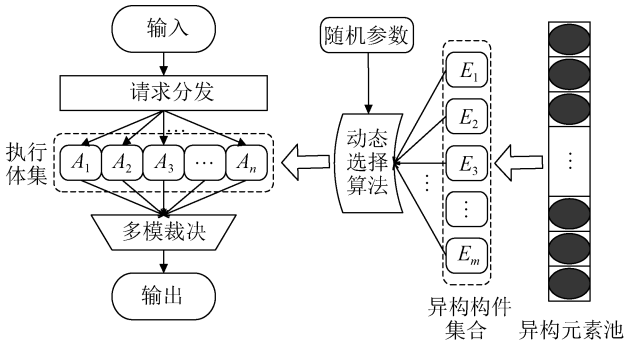

图 2 DHRA 模型

Figure 2 DHRA Model

DHRA 在保障执行体结构具备可重组、可重建、可重定义等动态化、异构化的结构变换要素的基础上, 通过构造“去协同化环境”[3-4], 从异构元素池选取异构元素, 组合生成 m 个具有相同作用和功能的异构构件, 进一步通过动态选择模块, 选择出 n 个异构构件作为执行体, 并按照这 n 个执行体的输出结果进行差异性判决, 屏蔽漏洞威胁, 使攻击者在时空维度上难以再现攻击场景。

2.3 多余度裁决

拟态防御架构通过多样化与随机化方法, 在时空维度上产生的不确定性, 在时间上以动态性呈现, 在空间上则以异构性呈现。其中, 多余度裁决方法在保证系统中不缺失动态性、异构性和冗余性的前提下, 比较响应输出, 得相对正确的响应结果屏蔽漏洞威胁, 反馈调节系统进行动态变换和系统重构, 是拟态防御架构的关键组成部分, 如图 3 所示。

多余度裁决更多的被视为一种响应比较的策略应用在 DHRA 中, 且在 DHRA 中要求了执行构件严格的去关联化或协同化, 所以对于多余度裁决并不会因此带来额外的安全问题。多余度裁决方法具体到不同的应用场景, 多余度裁决的实现方法会有一定的差异, 常用的裁决方法包括全体一致裁决[31-32], 大数裁决算法[33-34], 最大近似裁决[35], 基于历史信息的加权裁决[36-37]等。

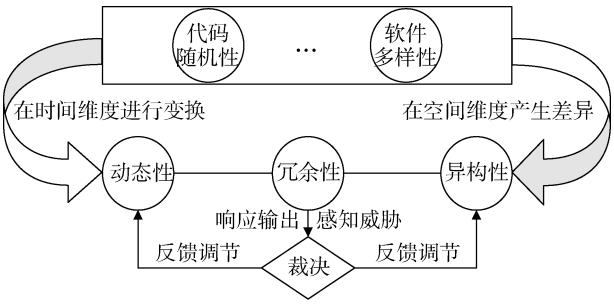

图 3 多余度裁决方法

Figure 3 The Way of Redundancy Adjudication

3 多余度裁决模型

3.1 符号说明

为便于后续描述, 给出如表 1 所示的符号定义。

表 1 本文用到的符号定义

Table 1 Symbolic definition

符号	定义说明
i, j	通用变量
m	系统总体构件个数 ($m \geqslant 0$)
n	系统执行体个数 ($0 < n \leqslant m$)
k	响应裁决输出一致的个数 ($\frac{n}{2} < k \leqslant n$)
λ_i	系统失效率 ($0 \leqslant \lambda_i \leqslant 1$)
t	执行时间 ($t_i \geqslant 0$)
u_1	系统自修复率 ($0 \leqslant u_1 \leqslant 1$)
u_2	系统离线修复率 ($0 \leqslant u_2 \leqslant 1$)
VUL_i	异构构件 $E_i\,(1 \leqslant i \leqslant m)$ 上漏洞的集合
$VUL = \bigcup_{i=1}^{m} VUL_i$	所有异构构件漏洞的集合
$H_1, H_2, \cdots, H_w$	w 个的异构元素
$E_1, E_2, \cdots, E_m$	w 个异构元素组合生成 m 个功能等价的异构构件
$A_1, A_2, \cdots, A_n$	从 m 个构件 E_i 中随机选择的 n 个执行体
$\Delta_{security}$	响应裁决相对单一构件响应结果的安全增益
$\Delta_{overhaed}$	响应裁决相对单一构件响应速率的系统开销
$\Delta_{maintenance}$	响应裁决感知到威胁后系统恢复能力

3.2 前提假设

针对拟态防御的多余度裁决方法的安全增益、系统开销和恢复能力进行建模分析。为便于对多余

度裁决模型的部署架构和风险进行分析与评估, 给出以下前提假设。

前提 1. 冗余状态下的数据机密性的保证, 一方面可以采用将数据与架构分离的方法来避免数据的冗余构造, 另一方面可以采用数据拟态化的方法来隐匿关键数据, 从而保障数据机密性不受影响。本文方法中不考虑模型对数据机密性的影响, 以更清晰的描述模型构建和评估方法。

前提 2. 在多状态服务场景下, 系统服务状态复杂多变。因此, 为了保证系统动态变换后的运行稳定性和裁决的准确性, 需要保持运行状态的上下文信息的存储和载入一致性。本文方法默认裁决过程中系统的运行状态具有一致性, 不考虑维护状态一致性的系统开销。

假设 1. 由于引入多构件, 系统的输入输出部件变得复杂, 为便于分析, 在本文中假设对于输入和输出部件的不会对系统安全增益的造成影响。

假设 2. 为便于分析, 假设执行体中任一漏洞遭受攻击成功则导致执行体失效, 且这种失效在时间维度上可以看作是均匀的随机事件。

假设 3. 为便于分析, 在简化模型中, 假设对系统中不同执行体的攻击致失效率是相同的, 且两个执行体间的失效是相互独立的。

假设 4. 为便于分析, 在简化模型中, 假设将比较操作的开销, 作为系统的全部开销。

假设 5. 为便于分析, 在简化模型中, 假设系统的构件集足够大且自修复率与离线修复率相同。

3.3 相关定义

令 $Set_H=\{H_1,H_2,\cdots,H_w\}$ 是所有用于组成系统构件体的异构元素之集, $Set_E=\{E_1,E_2,\cdots,E_m\}$ 是所有异构构件之集, $\mathrm{Set}_A=\{A_1,A_2,\cdots,A_n\}$ 是所有在线工作的异构执行体之集。对于 Set_E 中任意的异构构件 $E_i(1\leqslant i\leqslant m)$ 都由 Set_H 中若干异构元素构成。对于 Set_A 中的任意异构执行体 A_i 都由 Set_E 中的异构构件动态调度得到的。令构件 $E_i(1\leqslant i\leqslant m)$ 上漏洞之集记为 VUL_i, 所有构件漏洞之集记为 $VUL=\bigcup_{i=1}^{m}VUL_i$。在此基础上, 给出如下若干定义。

定义 1. 共生漏洞.

VUL_i 与 VUL_j 相互之间的重合部分称为 VUL_i 与 VUL_j 的共生漏洞。对于两个执行体间的共生漏洞, 将其称为 2 阶共生漏洞。以此类推, 将 k 个执行体间的漏洞称为 k 阶共生漏洞。

定义 2. *k/n* 裁决.

对于 n 个执行体的处理结果进行比较, 如果有至少 k 个执行体的输出结果一致, 则按照 k 个执行体的处理结果, 作为相对正确结果进行输出。将 *k/n* 视为裁决方法的严格程度。对于执行结果进行响应裁决, 如果执行结果完全一致, 则视为正常状态; 如果执行结果不一致, 则视为异常状态。可以看出, 如果系统中包含的最大共生漏洞阶数是小于 k 的, 那么采用 k 阶一致裁决方法能够感知到漏洞威胁; 如果系统中存在 k 阶或 k 阶以上共生漏洞, 那么采用 k 阶一致裁决的方法将不再能感知到该漏洞威胁, 产生拟态逃逸[2]现象。

定义 3. 系统失效.

对于执行体而言, 当攻击者通过某种攻击手段对执行体中的漏洞攻击成功时, 则认为系统失效。将系统失效这一事件视为服从失效率为 λ 的指数分布, 其中 λ 表示不同攻击者对不同漏洞威胁攻击导致系统失效能力, 代表致攻击致失效的难度。概率密度用 $v(t)$表示。

$$v(t)=\lambda e^{-\lambda t}$$

通过 $v(t)$可以很好的描述攻击者进行攻击导致系统失效的变化速率。当攻击者在初始时刻由于对于系统状况并不了解, 对于漏洞的利用导致系统失效的概率为零。但是攻击者在开始的一段时间内能够迅速了解系统状况, 系统失效概率迅速增加。随着时间的增长, 攻击者已经基本了解了系统的状况, 对于系统的了解逐渐变缓, 系统的失效概率慢慢趋近于一。

定义 4. 安全增益.

相对于单一执行过程, 拟态防御架构中引入异构多执行过程, 通过并行的执行和响应裁决的方法, 使得系统失效变得更加困难。将单个执行体的输出结果与经过拟态裁决后的系统输出进行对比, 将二者在面对攻击情况下保持正常运行状态的概率比值来刻画安全增益。记单个执行体失效概率为 F_i, 拟态裁决后的失效概率为 F_{judge}, 则安全增益 $\Delta_{security}$ 定义为:

$$\Delta_{security}=\frac{1-F_{judge}}{1-\min\limits_{1\leqslant i\leqslant n}(F_i)}$$

定义 5. 系统开销.

由于引入多异构执行体并行执行, 每个执行体的处理过程都需要消耗不同的响应时间, 导致拟态系统整体的响应时间要符合异构执行体中响应时间最慢的执行体, 服从“短板效应”。并且执行体的处理结果需要进行 *k/n* 裁决, 也需要一定的处理时间。此外, 在处理过程中还需消耗一部分时间用来对执行体进行动态调度。因此采用拟态防御架构的执行时间必然要大于单个执行体执行的执行时间。记单个执行体的执行时间为 t_i, 裁决时间为 t_{judge}, 调度

时间为 $t_{dispatch}$，则记系统开销 $\Delta_{overhaed}$ 为:

$$\Delta_{overhaed} = \max_{1\leqslant i\leqslant n} t_i + t_{judge} + t_{dispatch}$$

定义 6．恢复能力.

当系统通过拟态裁决感知到攻击威胁时, 执行体会通过重构、重组、重建异构构件的方法进行系统恢复。然而, 并不是所有的情况系统都能够进行自身修复, 部分持久化攻击威胁需要人为的进行错误排查。根据同类型错误的平均出现时间, 来区分自修复与离线修复。如果在时间 t 内, 同类型错误不在出现, 则系统能够自行修复, 否则需要人为的进行错误排查。若执行体自身能够在时间 t 内立即修复, 则不需要进行离线修复; 若执行体失效后不能在 t 内进行修复, 则在异构构件集合中选取异构构件上线执行服务, 替换已经失效的执行体。同时, 已失效的执行体离线修复, 修复成功后重新加入异构构件集合中, 等待再次上线。将这一过程用 4 个状态进行描述。状态 S_0 表示执行体处于正常服务状态; 状态 S_1 表示执行体处于失效状态不需要修复; 状态 S_2 表示执行体处于失效状态需要修复; 状态 S_3 表示执行体处于离线待服务状态。

记 $P_{S_0}(t)$、$P_{S_1}(t)$、$P_{S_2}(t)$、$P_{S_3}(t)$分别为 S_0、S_1、S_2、S_3 在 t 时刻的状态转移概率, 当单个执行体失效时, 将正常服务状态和离线待服务状态与执行体失效修复状态的差值作为单个执行体的恢复能力。进一步分析, 当多个执行体失效时, 分为自修复和离线修复两种状态讨论。只需自修复的执行体, 不需要离线执行体的替换, 所以 n 个在线服务的执行体都有自我修复的能力。而需要离线修复的执行体, 若处于离线待服务状态的执行体个数小于在线服务的执行体个数, 则最多能同时修复 $m–n$ 个执行体; 若处于离线待服务状态的执行体个数大于在线服务的执行体个数, 则最多能同时修复 n 个执行体。于是在感知到威胁后系统的恢复能力 Δ_{repair} 为:

$$\Delta_{repair} = \begin{cases} \sum_{i=1}^{n} P_{iS_0}(t) + \sum_{i=1}^{m-n} P_{iS_3}(t), & m < 2n \\ \sum_{i=1}^{n} P_{iS_0}(t) + \sum_{i=1}^{n} P_{iS_3}(t), & m \geqslant 2n \end{cases}$$

3.4 安全增益

本节根据定义 4 中定义的安全增益进行分析。对于整个拟态防御系统而言, 当多个执行体的漏洞被同时利用时, 则认为系统失效。根据在拟态防御系统中采用的 k/n 裁决机制, 可将多余度裁决模型的失效概率视为至少 k 个执行体失效的概率, 讨论由异构冗余构造和拟态裁决机制带来的安全增益。欲求至少 k 个执行体失效的概率, 需先求解恰有 k 个执行体失效的概率。为了便于书写, 引入如下记号:

$$W(1) = F(\sum_{i=1}^{m} |A_i|)$$

$$W(2) = F(\sum_{1\leqslant i<j\leqslant n} |A_i \cap A_j|)$$

$$W(3) = F(\sum_{1\leqslant i<j\leqslant k\leqslant n} |A_i \cap A_j \cap A_k|)$$

……

$$W(n) = F(|A_1 \cap A_2 \cap \cdots \cap A_n|)$$

对于计算满足某些性质的组合计数问题, 使用容斥原理能够很好的解决问题。容斥原理是一个古老而又简便的工具, 将现代组合计数手段与其相结合, 衍生出许多新意。广义容斥原理作为容斥原理的变形和推广, 能够很好的解决恰有 k 种性质的问题。设恰有 k 个执行体失效的概率为 $M(k)$, 根据广义容斥原理可知, 则可将 $M(k)$ 表示为关于 W 的通项公式:

$$M(k) = \sum_{i=0}^{n-k} (-1)^i \binom{k+i}{k} W(k+i)$$

于是有至少有 k 个执行体失效的概率为:

$$F_{\text{mimic}} = \sum_{i=k}^{n} M(i) = \sum_{i=k}^{n} \sum_{j=0}^{n-k} (-1)^j \binom{i+j}{i} W(i+j)$$

则安全增益为:

$$\Delta_{security} = \frac{1 - \sum_{i=k}^{n} \sum_{j=0}^{n-k} (-1)^j \binom{i+j}{i} W(i+j)}{1 - \min_{1\leqslant i\leqslant n} (F_i)}$$

进一步分析, 假设对系统中不同执行体的攻击致失效率是相同的, 即 $F_i = F_j = F$。且两个执行体间的失效是相互独立, 即 $F_{ij} = F_i \cdot F_j = F^2$, $(i \neq j)$。于是有恰有 k 执行体失效的概率为:

$$M(k) = \binom{k}{n} F^k \cdot (1-F)^{n-k}$$

于是有至少 k 个执行体失效的概率为:

$$F_{\text{mimic}} = \sum_{i=k}^{n} M(i) = \sum_{i=k}^{n} \binom{i}{n} F^i \cdot (1-F)^{n-i}$$

则安全增益为:

$$\Delta_{security} = \frac{1 - \sum_{i=k}^{n} \binom{i}{n} F^i \cdot (1-F)^{n-i}}{1-F}$$

若取 $\lambda_1=\lambda_2=\cdots\lambda_n=\lambda$, 取 $k=n$ 时, 由定义 3 给出单个执行体的失效概率, $F_i = 1 - e^{-\lambda_i t}$, 则系统安全增益为:

$$\Delta_{security}=\frac{1-(1-e^{-\lambda t})^n}{e^{-\lambda t}},n\geqslant 1$$

3.5 系统开销

根据定义 5 中定义的系统开销对多余度裁决模型的执行效率进行分析。为便于分析, 先不考虑动态调度带来的系统开销, 分析引入 *k/n* 裁决机制所带来的系统开销, 对 *n* 个执行体的输出结果进行比较, 对异常状态的感知是整个系统执行安全的关键方法。对于执行体 *A* 的执行结果, 为其维护一个 *Ja* 结构数组包括 *num* 和 *val*, 分别表示状态个数和状态值。在表 2 中给出多余度裁决算法的基本流程。

在多余度裁决算法中, 比较操作是耗时最长的操作, 其他操作相对于比较来说可以忽略不计。对算法的执行结果进行分析可知: 最好情况, 每个执行体的执行结果都相等, 则只需要进行 *n*–1 次比较操作; 最坏情况, 每个执行体的执行结果都不相等, 则需要进行 $n(n-1)/2$ 次比较操作。进一步分析, 在某些特殊执行过程中, 执行结果的状态空间只会产生正确的和错误的两种情况, 那么算法的执行结果则只需进行 *n*–1 次比较操作。记比较方法的执行时间为 $t_{compare}$, 则 $t_{judge}=(n-1)t_{compare}+O(t)$, 所以可以得出结论, 在相同的比较算法下, 有 $\Delta_{overhaed}$ 与 *n* 成正相关。

表 2 *k/n* 裁决

Table 2 *k/n* Adjudication

```
Judge(A)
A[n]=A1A2…An
Ja[n].num←{0}  Ja[n].val←{NULL}
Ja[1].val = A[1]
Ja[1].num++
FOR i = 2 to n
  FOR j = 1 to i
    IF Ja[j].num == 0 THEN
      Ja[j].val = A[i]
      Ja[j].num++
      BREAK
    END IF
    IF Compare( Ja[j].val ,A[i] ) THEN
      Ja[j].num++
      BREAK
    END IF
  END FOR
END FOR
FOR i=1 to n
  IF Ja[i].num > k THEN
    RETURN TRUE
  ELSE
     RETURN FALSE
  END IF
END FOR
```

3.6 恢复能力

本节根据定义 6 中定义的恢复能力对多余度采掘模型进行威胁感知后进行系统恢复的情况进行分析。当系统失效后, 系统会进入自修复或进入离线修复状态。设系统失效后进行自行修复的概率为 u_1, 离线修复率后能够正常服务的概率为 u_2, 则将多余度裁决模型感知到威胁后对执行体状态进行系统恢复这一过程用状态转移图进行描述, 如图 4 所示。

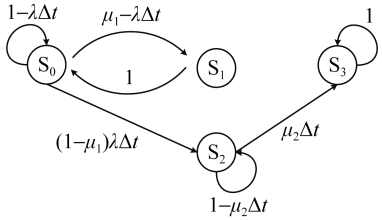

图 4 系统恢复状态转移图

Figure 4 System recovery state transition diagram

设 $P_{ij}(t)$表示从状态 *i* 到状态 *j* 的转移概率, 则根据系统恢复的状态转移图所构造的的状态转移矩阵为:

$$P(t)=\begin{bmatrix}p_{00}(t) & p_{01}(t) & p_{02}(t) & p_{03}(t)\\ p_{10}(t) & p_{11}(t) & p_{12}(t) & p_{13}(t)\\ p_{20}(t) & p_{21}(t) & p_{22}(t) & p_{23}(t)\\ p_{30}(t) & p_{31}(t) & p_{32}(t) & p_{33}(t)\end{bmatrix}$$

该矩阵满足柯尔莫哥洛夫向后方程并且满足条件,

$$\begin{cases}\dfrac{\mathrm{d}P(t)}{\mathrm{d}t}=P(t)\boldsymbol{Q}\\ P(0)=\boldsymbol{E}\end{cases}$$

其中, $\boldsymbol{E}$ 为 4 阶单位矩阵, $\boldsymbol{Q}$ 为马尔科夫过程转移速率矩阵。由连续马尔科夫正则性条件给出矩阵 $\boldsymbol{Q}$ 的求解方法,

$$\begin{cases}\lim\limits_{\Delta t\to 0}\dfrac{1-p_{ii}(\Delta t)}{\Delta t}=q_{ii}\\ \lim\limits_{\Delta t\to 0}\dfrac{p_{ij}(\Delta t)}{\Delta t}=q_{ij},i\neq j\end{cases}$$

于是有,

$$\boldsymbol{Q}=\begin{bmatrix}-\lambda & \mu_1\lambda & (1-\mu_1)\lambda & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & -\mu_2 & \mu_2\\ 0 & 0 & 0 & 0\end{bmatrix}$$

代入参数 u_1 、u_2 求解微分方程:

$$\begin{cases}\dfrac{\mathrm{d}P_{S0}(t)}{\mathrm{d}t}=-\lambda P_{S0}(t)\\ \dfrac{\mathrm{d}P_{S1}(t)}{\mathrm{d}t}=\mu_1\lambda P_{S0}(t)\\ \dfrac{\mathrm{d}P_{S2}(t)}{\mathrm{d}t}=(1-\mu_1)\lambda P_{S0}(t)-\mu_2 P_{S2}(t)\\ \dfrac{\mathrm{d}P_{S3}(t)}{\mathrm{d}t}=\mu_2 P_{S2}(t)\end{cases}$$

由初始条件 $P_{S_0}(0)=1, P_{S_1}(0)=P_{S_2}(0)=P_{S_3}(0)=0$, 解得,

$$P_{S_0}(t)=e^{-\lambda t},$$

$$P_{S_1}(t)=\mu_1(1-e^{-\lambda t}),$$

$$P_{S2}(t)=\frac{(1-\mu_1)\lambda}{\lambda-\mu_2}(e^{-\mu_2 t}-e^{-\lambda t}),$$

$$P_{S3}(t)=\frac{(1-\mu_1)\mu_2}{\lambda-\mu_2}e^{-\lambda t}-\frac{(1-\mu_1)\lambda}{\lambda-\mu_2}e^{-\mu_2 t}+1-\mu_1。$$

为了便于计算, 取 $u_1=u$, $u_2=u\lambda$, $\lambda_1=\lambda_2=\cdots\lambda_n=\lambda$, 讨论当异构构件集合足够大时, 可以保证 $n\ll m$, 有 $\Delta_{repair}=n(e^{-\lambda t}+\mu e^{-\lambda t}-e^{-\mu\lambda t}+1-\mu)$。

4 模型分析

4.1 仿真分析

根据对多余度裁决模型的安全增益、系统开销和恢复能力的分析, 讨论在不同失效概率和不同余度的条件下, 安全增益、执行效率和恢复能力这三个要素的相互影响。取系统开销的倒数作为执行效率, 取 λ=1、u=0.5 的简化模型, 则有,

$$\begin{cases}\Delta_{security}=\dfrac{1-(1-e^{-t})^n}{e^{-t}}\\ 1/\Delta_{overhead}=1/(n-1)*t\\ \Delta_{repair}=n(1.5e^{-t}-e^{-t/2}+0.5)\end{cases}$$

根据简化模型参数, 在 Matlab 上进行了仿真实验, 实验结果如图 5 所示。

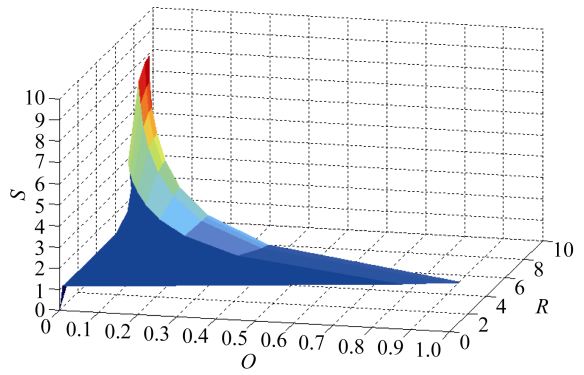

图 5 安全增益、系统开销和恢复能力的变化关系

Figure 5 The model indicators' potential relationship

在不同余度的裁决模型下, 以不同的失效概率, 比较时间和恢复时间构造的防御系统, 系统的安全增益、执行效率和恢复能力之间存在着相互影响。当系统的安全增益增加, 系统执行效率减小, 系统恢复能力增加。这说明了引入多余度裁决, 在提高系统防御能力和恢复能力的同时, 会产生一定的性能损耗。

4.2 部署风险

在简化模型中安全增益的计算是以 $k=n$ 作为前提条件, 系统采用完全一致裁决的策略保证了引入多余度裁决方法防御能力的提升。在实际部署中, 如果系统能够保证严苛的相异性要求, 系统中几乎不存在共生漏洞, 裁决阈值的选取则不需要很严格。而在实际设计中如果降低了相异性设计要求, 那么系统中出现共生漏洞的概率大大增加, 而此时采取不同余度的裁决方法会对系统安全增益产生不同的影响, 部署不当则会产生拟态逃逸现象。

根据拟态防御 web 服务器的设计与实现方法[8], 以异构的 web 服务器软件为例, 对多余度裁决模型的部署风险分析。为便于分析, 不考虑系统服务状态对系统攻击产生影响且当 web 服务器软件选用 Apache、IIS 和 Nginx 时, 对于异构 web 服务器的攻击是相互独立的。讨论执行构件的相异性和不同余度的裁决方法对系统的安全增益的影响。分别讨论执行体组合选取完全异构、部分异构和完全同构情况下, 系统的安全增益的变化。

在完全同构的情况下, 可将每个执行体视作等价执行体, 若单个执行体失效则相当于整个系统失效, 于是有,

$$\Delta_{security}=\frac{1-F(|A_1\cap A_2\cap\cdots\cap A_n|)}{1-\min\limits_{1\leqslant i\leqslant n}(F_i)}=\frac{1-F_i}{1-F_i}=1。$$

在完全异构的情况下, 系统中的共生漏洞几乎为零, 可将异构执行体视作完全不相交, 则有,

$$\Delta_{security}\approx\frac{1}{1-\min\limits_{1\leqslant i\leqslant n}(F_i)}>1。$$

在部分同构的情况下, 单个执行体失效概率会影响与其同构的组合, 系统中存在不可忽略的共生漏洞, 需要考虑不同余度的裁决方法对系统安全增益的影响。在完全一致裁决的情况下有,

$$F(|A_1\cap A_2\cap\cdots\cap A_n|)\leqslant\min_{1\leqslant i\leqslant n}(F_i),$$

$$\Delta_{security}=\frac{1-F(|A_1\cap A_2\cap\cdots\cap A_n|)}{1-\min\limits_{1\leqslant i\leqslant n}(F_i)}\geqslant 1。$$

在 k 阶一致表决情况下, 安全增益则不能确定。以 2/3 裁决情况下部分同构所带来的问题为例, 首先计算出在 2/3 裁决情况下的安全增益为:

$$\Delta_{security_{2/3}}=\frac{1-(F_1F_2+F_1F_3+F_2F_3-2F_1F_2F_3)}{1-\min\limits_{1\leqslant i\leqslant n}(F_i)};$$

如果系统中只选取了 1 个 Apache 和 2 个 Nginx 服务器作为执行构件, 那么安全增益为:

$$\Delta_{security_{2/3}}=\frac{1-(F_AF_N+F_AF_N+F_N-2F_AF_N)}{1-\min\limits_{1\leqslant i\leqslant n}(F_i)}=\frac{1-F_N}{1-\min\limits_{1\leqslant i\leqslant n}(F_i)}\leqslant 1$$;

此时 Nginx 服务器的失效概率将作为系统的失效概率。同理可知, 选取了 2 个 Apache 和 1 个 Nginx 服务器, 会以 Apache 服务器的失效概率作为系统的失效概率, 安全增益为:

$$\Delta_{security_{2/3}}=\frac{1-F_A}{1-\min\limits_{1\leqslant i\leqslant n}(F_i)}\leqslant 1\text{。}$$

由上述分析可知, 在部分同构的情况下, 不完全一致裁决的方法是危险的。所以在实际部署中, 应尽量避免执行构件中出现的同构成分的情况下, 同时对于出现部分同构的情况, 应尽量使用完全一致裁决的方法, 以保证拟态防御多余度裁决模型能够最大程度发挥其防御效果。

5 实验评估

根据拟态防御 web 服务器的实现与测试方法[8-9], 以异构的 web 服务器软件为例, 对多余度裁决模型部署及其风险进行测试与评估。web 服务器的文件解析漏洞经常被攻击者作为网络渗透的突破口, 而异构服务器的文件解析漏洞的利用方法是不一致的。因此, 本文选取 Apache 文件解析问题和 Nginx 配置 fastcgi 解析 PHP 时存在的文件解析问题来验证引入多余度裁决模型系统的防御能力、系统开销、恢复能力以及可能出现的部署风险。预先上传包含恶意 PHP 代码的 test.jpg 和 test.php.rar.rar.rar 文件。对于 Apache 服务器文件解析是从后往前执行的, 构造 http://127.0.0.1/path/test.php.rar.rar.rar, 由于服务器不能识别后缀为.rar 的文件, 所以会一直遍历后缀直到.php 结束, 将 test.php.rar.rar.rar 作为 PHP 执行; 对于 Nginx 服务器构造 http://127.0.0.1/path/test.jpg/noexist.php, 由于 noexist.php 是不存在的, 服务器会递归查询确认文件的合法性, 最终将 test.jpg 当做 PHP 进行解析。

实验环境选取三台异构的 web 服务器, Ubuntu+Apache, CentOS+Nginx, WinServer+IIS, 并已关闭入侵检测和访问控制等传统的 web 服务安全机制[38], 同时对操作系统和服务器软件进行了优化[39]。分别选取不同构造方法的多余度裁决模型, 对系统的防御能力、运行效率和失效恢复进行测试。实验结果, 如表 3 所示。

讨论采用不同余度的裁决方法和不同类型的服务器对于文件解析漏洞的防御能力影响。当 n=3, k=2 时, 采用 2 台或 3 台 Apache 服务器都会执行用例 1 的预置脚本; 采用 2 台或 3 台 Nginx 服务器则都会执行用例 2 的预置脚本。当 n=3, k=3 时, 采用 2 台 Apache 或 2 台 Nginx 服务器, 预置脚本都不会执行。当 n=2, k=1 时, 采用 1 台 Apache 和 1 台 Nginx 服务器混合的策略, 两个预置脚本都能执行, 而单独使用 2 台 Apache 或 Nginx 服务器则只能执行其中一个预置脚本。这说明了同时采用不完全一致裁决方法和部分同构的异构性构造策略构造的拟态防御系统是有风险的, 会产生拟态逃逸现象, 严重时可能会产生安全增益损失的情况。对于不完全一致裁决的极限情况 k=n/2 时, 完全异构的异构性构造策略也是危险的, 所以在多余度裁决模型中要求了 k>n/2。

讨论采用不同余度的裁决方法和不同类型的服务器对于系统执行效率影响。对比响应时间, 将响应时间进行归一化处理, 以最小执行时间为单位时间, 得到响应速率比。首先, 对比选取不同类型服务器的响应速比。在 n 固定的情况下, 不同执行体的响应时间略有不同, 选取了 IIS 服务器的系统响应时间相对较长。 然后, 讨论多余度裁决机制对系统执行效率的影响。选取同一类型的服务器进行对比, 当 n 从 1 增长到 3 时, 由于裁决带来的系统开销, 响应时间也会随之增长。最后, 以 n=1 时的测试数据为基准, 与 n>1 时测试数据进行比较, 可知在执行体组合的服务质量差异较小的情况下, 多余度裁决机制是影响系统执行效率的首要因素。

讨论采用不同余度的裁决方法和不同类型的服务器对于系统恢复情况。为了能够更好的分析系统恢复的情况, 在实验中取 m=2n, 保证在最坏情况下, 三个执行服务器同时失效, 系统也能正常运行。可以发现, 当系统感知到漏洞威胁并屏蔽了代码执行的测试用例, 系统都能够进行恢复。并且, 在采用完全一致裁决的情况下, n 越大系统能够恢复的情况数越多, 系统的恢复能力越强; 在 n 相同的情况下, 裁决策略越严格, 系统能够恢复的情况数越多, 系统的恢复能力越强。

6 总结

本文根据拟态防御架构和多余度裁决模型之间的关系, 针对拟态防御的多余度裁决模型, 对系统的防御能力、运行效率和恢复能力三方面进行分析。并对结合 web 服务器进行风险评估, 并给出部署意见。最后对多余度裁决模型的变化趋势和部署风险行了实验评估。在实验分析中只选取了 web 服务软

表 3　实验评估

Table 3　Experimental Evaluation

m	*n*	*k*	服务器			测试用例				响应速率比
			Apache	Nginx	IIS	../test.php.rar.rar.rar		./test.jpg/noexist.php		
						代码执行	系统恢复	代码执行	系统恢复	
6	3	2	**3**	**0**	**0**	√	×	-	-	**3.86**
			0	**3**	**0**	-	-	√	×	**4.43**
			2	**1**	**0**	√	×	×	√	**5.29**
			1	**2**	**0**	×	√	√	×	**5.86**
			1	**1**	**1**	×	√	×	√	**9.57**
		3	**3**	**0**	**0**	√	×	-	-	**4.71**
			0	**3**	**0**	-	-	√	×	**5.86**
			2	**1**	**0**	×	√	×	√	**5.57**
			1	**2**	**0**	×	√	×	√	**5.29**
			1	**1**	**1**	×	√	×	√	**9.43**
4	2	1	**2**	**0**	**0**	√	×	-	-	**3.00**
			0	**2**	**0**	-	-	√	×	**3.85**
			1	**1**	**0**	√	×	√	×	**6.71**
			2	**0**	**0**	√	×	-	-	**2.14**
		2	**0**	**2**	**0**	-	-	√	×	**4.71**
			1	**1**	**0**	×	√	×	√	**6.14**
2	1	1	**1**	**0**	**0**	√	×	-	-	**1.00**
			0	**1**	**0**	-	-	√	×	**1.50**
			0	**0**	**1**	-	-	-	-	**3.85**

件的基本漏洞进行了实验分析, 未考虑涉及复杂状态集服务的漏洞攻击分布的情况以及在采用多个层次组合的异构化情况下, 各个层次网络渗透攻击之间的相互依赖关系, 在下一步工作中继续完善。

致谢　本文工作受到国家重点研发计划网络空间安全专项(No. 2017YFB0803201), 上海市科学技术委员会科研计划项目(No. 16DZ1120502)资助。

参考文献

[1] Council S T. "Federal Cybersecurity Research and Development Strategic Plan," CreateSpace Independent Publishing Platform, 2016.

[2] J.X. Wu, "Research on Cyber Mimic Defense," *Journal of Cyber Security*, vol. 1, no. 4, pp.1-10(in Chinese), 2016.
(邬江兴, "网络空间拟态防御研究", *信息安全学报*,2016,1(04):1-10。)

[3] J.X. Wu, "Cyber space Mimic Security Defense," *Secrecy Science and Technology*, vol. 10, no. 1, pp. 4-9(in Chinese), 2014.
(邬江兴, "网络空间拟态安全防御", *保密科学技术*, 2014, 10(1): 4-9。)

[4] J.X. Wu, "Meaning and Vision of Mimic Computing and Mimic Security Defense," *Telecommunications Science*, vol. 30, no. 7, pp. 1-7(in Chinese), 2014.
(邬江兴, "拟态计算与拟态安全防御的原意和愿景", *电信科学*, 2014, 30(7): 1-7。)

[5] J.X. Wu, F. Zhang and X.G. Luo, "Mimic computing and Mimic Security Defense," *Communications of the CCF*, vol. 11, no. 1, pp. 8-14(in Chinese), 2015.
(邬江兴, 张帆, 罗兴国, "拟态计算与拟态安全防御" , *中国计算机学会通讯*, 2015, 11(1): 8-14。)

[6] X.M. Si, W. Wang, J.J.Zeng, B.C.Yang, G.S.Li, C.Yuan and F.Zhang, "A Review of the Basic Theory of Mimic Defense," *Engineering Sciences*, vol. 18, no. 6, pp. 62-68(in Chinese), 2016.
(斯雪明,王伟,曾俊杰,杨本朝,李光松,苑超,张帆, "拟态防御基础理论研究综述" , *中国工程科学*, 2016, 18(6): 62-68。)

[7] H.C.Hu, F.C.Chen and Z.P.Wang, "Performance Evaluation on DHR for Cyberspace Mimic Defense," *Journal of Cyber Security*, vol. 1, no. 4, pp.40-51(in Chinese), 2016.
(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", *信息安全学报*, 2016, 1(4): 40-51。)

[8] Q.Tong, Z.Zhang, W.H.Zhang and J.X.Wu, "Design and Implementation of Mimic Defense Web Server," *Journal of Software*, vol.28, no.4, pp.883-897(in Chinese), 2017.
(仝青,张铮,张为华,邬江兴, "拟态防御 Web 服务器设计与实现", *软件学报*, 2017, 28(4): 883-897。)

[9] Z.Zhang, B.L.Ma and J.X.Wu, "The Test and Analysis of Prototype of

Mimic Defense in Web Servers," *Journal of Cyber Security*, vol. 2, no. 1, pp.13-28(in Chinese), 2017.
(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", *信息安全学报*, 2017, 2(1): 13-28。)

[10] H.L.Ma, P.Yi, Y.M.Jiang and L.He, "Dynamic Heterogeneous Redundancy based Router Architecture with Mimic Defenses," *Journal of Cyber Security*, vol. 2, no. 1, pp.29-42(in Chinese), 2017.
(马海龙,伊鹏,江逸茗,贺磊, "基于动态异构冗余机制的路由器拟态防御体系结构", *信息安全学报*, 2017, 2(1): 29-42。)

[11] H.L.Ma, Y.M.Jiang, B,Bai and J.H.Zhang, "Test and Analyses for Mimic Defense Ability of Routers," *Journal of Cyber Security*, vol. 2, no. 1, pp.45-53(in Chinese), 2017.
(马海龙,江逸茗,白冰,张建辉, "路由器拟态防御能力测试与分析", *信息安全学报*, 2017, 2(1): 43-53。)

[12] H.C.Yi, "Security Method and Device for DNS Recursive Server," CN 106961422 A (in Chinese), 2017.
(扈红超, "一种 dns 递归服务器的拟态安全方法及装置", CN 106961422 A. 2017。)

[13] S.Wei, H.Yu, Z.Y.Gu and X.M.Zhang, "Architecture of Mimic Security Processor for Industry Control System," *Journal of Cyber Security*, vol. 2, no. 1, pp.54-73(in Chinese), 2017.
(魏帅,于洪,顾泽宇,张兴明, "面向工控领域的拟态安全处理机架构",*信息安全学报*, 2017, 2(1): 54-73。)

[14] C.X.Liu, X.S.Ji and J.X.Wu, "A Mimic Defense Mechanism for Mobile Communication User Data Based on MSSIDN Virtualization," *Journal of Computer* , (in Chinese) http://kns.cnki.net/kcms/detail/11.1826.TP.20170508.1441.002.html , May 8th ,2017.
(刘彩霞,季新生,邬江兴, "一种基于 MSISDN 虚拟化的移动通信用户数据拟态防御机制", *计算机学报,* 2017:1-14(2017-05-08)。http://kns.cnki.net/kcms/detail/11.1826.TP.20170508.1441.002.html)

[15] Littlewood, Bev, and L. Strigini, "Redundancy and Diversity in Security," *Computer Security - ESORICS 2004, European Symposium on Research Computer Security, Sophia Antipolis, France*, September 13-15, 2004, Proceedings (Vol.3193, pp.423-438). DBLP.

[16] Q.Tong, Z.Zhang and J.X. Wu, "The Active Defense Technology Based on the Software/Hardware Diversity," *Journal of Cyber Security*, vol. 2, no. 1, pp.1-12(in Chinese), 2017.
(仝青,张铮,邬江兴, "基于软硬件多样性的主动防御技术", *信息安全学报*, 2017, 2(1): 1-12。)

[17] Y. J. Zhang and J.M.Pang, "A New Compile-Time Obfuscation Scheme for Software Protection," *in 2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 1-5, 2016.

[18] J.M.Pang, Y.J.Zhang, Z.Zhang and J.X.Wu, "Applying a Combination of Mimic Defense and Software Diversity in the Software Security Industry," *Engineering Sciences*, vol. 18, no. 6, pp. 74-78(in Chinese), 2016.
(庞建民, 张宇嘉, 张铮, 邬江兴, "拟态防御技术结合软件多样化在软件安全产业中的应用", *中国工程科学*, 2016, 18(06): 74-78。)

[19] B. Baudry and M. Monperrus, (2015). "The multiple facets of software diversity: Recent developments in year 2000 and beyond," *Eprint Arxiv*, vol. 48, no.1, pp.1-26, 2014.

[20] Piao Y, Jung J, Yi J H, "Server‐based code obfuscation scheme for APK tamper detection," *Security & Communication Networks*, 2016, 9(6): 457-467.

[21] M. Franz, "E unibus pluram: massive-scale software diversity as a defense mechanism," *In Proceedings of the 2010 workshop on New security paradigms*, pp. 7-16, 2010.

[22] A. Avizienis and L. Chen, "On the implementation of N-version programming for software fault tolerance during execution," *In Proc. IEEE COMPSAC*, pp. 149-155,1977.

[23] C. Hewa Nadungodage, Y. Xia, P.S. Vaidya, et al., "Online multi-dimensional regression analysis on concept-drifting data streams," International Journal of Data Mining, vol. 6, no. 3, pp. 217-238, 2014.

[24] fVoas J, Ghosh A, Charron F, et al. "Reducing uncertainty about common-mode failures." *In Proc. IEEE Symp. Software Reliability Engineering (SRE'97)*, pp. 308-319, 1997.

[25] Stamp, M, "Risks of monoculture.," *Communications of the Acm*, vol. 47, no. 3, pp. 120-120, 2004.

[26] J.Han, B.Y.Zang, "Analyzing the Effectiveness of Software Diversity for System Security," *Computer Applications & Software*, 2010.

[27] W. Peng, F. Li, C.T. Huang, et al., "A moving-target defense strategy for cloud-based services with heterogeneous and dynamic attack surfaces", *2014 IEEE International Conference on Communications (ICC)*, IEEE, pp.804-809, 2014.

[28] A.K. Bangalore and A.K. Sood, "Securing web servers using self cleansing intrusion tolerance (scit)," *DEPEND'09*, Second International Conference on. IEEE, pp. 60-65, 2009.

[29] J.H. Lala, "Organically Assured & Survivable Information Systems (OASIS): Foundations of Intrusion Tolerant Systems," *IEEE Computer Society Press*, 2003.

[30] Y. Huang and A.K. Ghosh, "Introducing diversity and uncertainty to create moving attack surfaces for web services," *Moving Target Defense*, Springer New York, pp. 131-151, 2011.

[31] Levitin G, "Optimal structure of fault-tolerant software systems," *Reliability Engineering & System Safety*, vol. 89, no. 3, pp. 286-295, 2005.

[32] P. R. Lorczak, A. K. Caglayan, and D. E. Eckhardt, "A Theoretical Investigation of Generalized Voters for Redundant Systems," *in The Nineteen*th International Symposium on Fault-Tolerant Computing, pp. 444 – 451, 1989.

[33] Gersting J L, Nist R L, Roberts D B, et al, "A comparison of voting algorithms for n-version programming," *Twenty-Fourth Hawaii International Conference on System Sciences. IEEE,* vol. 2 pp. 253-262, 1991.

[34] G. Latif-Shabgahi and S. Bennett. "Adaptive majority voter: a novel voting algorithm for real-time fault-tolerant control systems," *EUROMICRO Conference, 1999. Proceedings*, pp.113-120,1999.

[35] Y.W. Leung, "Maximum likelihood voting for fault-tolerant software with finite output-space," *IEEE Trans. Reliability*, vol.44, no.3, pp.419–427, 1995.

[36] G. Latif-Shabgahi, J.M. Bass and S. Bennett, "History-based weighted average voter: a novel software voting algorithm for fault-tolerant computer systems" in *Proceedings. Of Parallel and Distributed Processing, Ninth Euromicro Workshop on. IEEE*, pp.402-409, 2001.

[37] C.T.OuYang, X.Wang, J.Zheng, "Adaptive Consensus Voting Algorithm," vol.. 38, no. 7, pp. 130–133, 2011.
(欧阳城添, 王曦, 郑剑, "自适应一致表决算法", *计算机科学*, 2011, 38(7): 130-133。)

[38] Curphey M, and Araujo R., "Web application security assessment tools," *IEEE Security & Privacy*, vol. 4, no.4, pp.32-41, 2006.

[39] Yao.Y and Xia.J, "Analysis and research on the performance optimization of Web application system in high concurrency environment," *Information Technology, Networking, Electronic and Automation Control Conference*, *IEEE.* pp. 321-326, 2016.

李卫超 于 2016 年在哈尔滨工业大学信息安全专业获得学士学位。现于信息工程大学计算机科学与技术专业攻读硕士学位。研究领域为网络安全。研究兴趣包括: 主动防御技术、网络体系结构。
Email: 010liweichao@163.com

王立群 于 2016 年在哈尔滨工业大学计算机科学与技术专业获得学士学位。现于信息工程大学计算机科学与技术专业攻读硕士学位。研究领域为网络安全。研究兴趣包括: 主动防御技术、网络体系结构。Email: standoutking@163.com

张铮 于 2006 年在信息工程大学计算机科学与技术专业获得博士学位。现任数学工程与先进计算国家重点实验室副教授。研究领域为网络安全、先进计算。研究兴趣包括: 主动防御技术、高性能计算。Email: ponyzhang@126.com

邬江兴 现任国家数字交换系统工程技术研究中心主任, 教授, 博导。研究领域为信息通信网络、网络安全。研究兴趣包括: 主动防御、交换技术与宽带信息网络、高效能计算。Email: 17034203@qq.com