文章编号: 2095-2163(2018) 04-0037-06 中图分类号: TP393.4 文献标志码: A

动态异构冗余的 Web 威胁感知技术研究

李卫超[1]，冯俊龙[2]

(1 数学工程与先进计算国家重点实验室，郑州 450001; 2 哈尔滨工业大学 计算机科学与技术学院，哈尔滨 150001)

摘 要: 威胁感知技术作为保障系统稳定运行的重要组成部分，在 Web 服务系统中得到了广泛的应用。针对动态异构冗余的 Web 服务环境，利用多余度裁决的方法感知威胁，使得系统具备良好的阻断攻击过程、感知漏洞威胁的能力。本文首先介绍了动态异构冗余架构及其 Web 威胁感知系统的设计方法。然后通过比较不同威胁感知系统在不同架构下的应用效果，阐述动态异构冗余的 Web 威胁感知系统的设计特点与难点。最后总结并展望动态异构冗余的 Web 威胁感知技术面临的挑战和发展前景。

关键词: 动态异构冗余架构; 威胁感知; 异构 Web 容器; Web 服务软件栈

Research on dynamic heterogeneity redundant Web threat awareness technology

LI Weichao[1], *FENG Junlong*[2]

(1 State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China;
2 School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

Abstract: Threat awareness technology, as an important part of ensuring the stable operation of the system, has been widely applied in the Web service system. According to the dynamic heterogeneous redundant (DHR) Web service environment, the system has a good ability to block the attack process and perceive the threat of vulnerability by using the method of redundancy adjudication to perceive the threat. First, the paper introduces the DHR structure and design method of the Web threat awareness system. Second, the paper explains the security design features and difficulties of the DHR Web threat awareness system by comparing the application effect of different DHR structure. Finally, the summarization and the development of the DHR Web threat awareness system are discussed.

Key words: dynamic heterogeneity redundant structure; threat awareness; heterogeneity Web container; Web service software stack

引言

随着互联网技术的迅速发展，网络已经成为人们生活中不可替代一部分。与此同时，快速、大规模的软件开发模式，导致网络应用环境极易出现漏洞威胁。而且，自动化攻击工具的出现，大大降低了攻击者的攻击成本，给网络空间安全带来了极大的挑战[1]。为了应对网络空间中潜在的安全威胁问题，一种可行的方法是通过利用多样化和随机化的方法，在网络空间基础设施中构建动态异构冗余(Dynamic Heterogeneity Redundant, DHR) 模型，改变了原有系统的单一确定易受攻击的状态，从而达到扰乱攻击流程，阻断攻击威胁的目的[2-3]。

Web 服务作为关键的网络空间基础设施服务之一，首当其冲面临攻击威胁。因此，一种基于 DHR 架构的 Web 服务系统就此产生[4]。近年来，DHR Web 服务系统的发展十分迅速，在异构性构造、多余度裁决和动态调度等技术方面进展突出。文献 [5] 中，阐述了 DHR Web 服务器的基本构造方法及性能指标。文献 [6] 中，提出了利用软、硬件的多样性来构造系统异构性的方法; 文献 [7] 中，提出了一种软件多样化编译的方法来抵御攻击威胁; 文献 [8] 中，提出了一种防御 SQL 注入攻击的异构标签化方法。文献 [9] 中，提出了一种相似度求解方法，增强了 DHR 架构系统的判别和决策能力; 文献 [10] 中，提出了一种动态调度的模型和方法来抵御持续性威胁。

1 DHR 架构技术

DHR 架构是 DHR Web 威胁感知系统的环境基础。通过利用 DHR 架构技术将动态性、异构性和冗余性等基本的内生安全属性导入网络空间的基

基金项目: 上海市科学技术委员会科研计划项目(16DZ1120502) 。

作者简介: 李卫超(1994-) ，男，硕士研究生，主要研究方向: 网络安全; 冯俊龙(1979-) ，男，硕士，工程师，主要研究方向: 网络安全。

收稿日期: 2018-06-13

础设施服务中，构建内生安全的网络架构模型，从而达到阻断攻击链、感知漏洞威胁的效果。

1.1 DHR 特性

DHR 架构具有内生安全的特点。因此，对于内生安全特性有着深刻的理解才能更好地理解 DHR 架构技术的内生安全机制。DHR 特性包括：异构性、冗余性、多样性、动态性、随机性等。基于此，可做研究解析分述如下。

(1) 异构性。是指具有相同功能作用的构件，在结构构造、处理过程和实现方法存在的差异性。文献[6]中，全面分析了多种软件漏洞，并测试了相异性对安全性的提升是有效的。文献[11]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。对于 2 个不同的处理构件，相互之间必然会存在一定的差异性。理论上来说，如果 2 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 2 个构件是不可能同时生效的。

(2) 冗余性。是指在系统中存在功能等价的构件同时工作。早在 1834 年，Lardner 就已经提出“通过多台相同的计算机处理同一运算，来检测运算结果的正确性，如果采用不同的运算方法进行计算，那么结果则更加可靠”的论题[12]。冗余性可以分为同构冗余和异构冗余。系统的冗余程度与系统的抗攻击能力没有直接的关联，对于同构冗余能够做到屏蔽构件自身执行产生的故障性错误，在系统容错方面起着重要的作用。对于异构冗余模型，除了能够达到容错的要求，还能够满足系统对容侵的需求，提高系统的抗攻击能力。

(3) 多样性。是指在一个工作系统中构件的丰富程度。Baudry 等人对软件多样性的多个方面，包括系统安全性等内容进行了全面的分析[13]。复杂的异构性和冗余性与系统的多样性是不可分离的。可以认为，系统中构件的差异程度越大，异构性和冗余性越强。而系统中的构件集就越丰富，多样性越强。因此，多样性常被用于异构性的构造方法之中。

(4) 动态性。是指具有相同功能作用的构件相互之间动态切换，达到让系统结构不确定的目的。Peng 等人在云服务环境下，利用虚拟化技术进行系统恢复的方法，实现了服务和流量的动态迁移[14]。Lew 等人应用了动态变迁技术实现了一种动态防御的安全防护系统[15]。动态变换要按照一定的策略，不定时地改变系统运行构件，降低了系统漏洞被发现的风险，同时对持续性攻击行为有着良好的防御效果。从某种程度上来说，系统的动态性可以被视为在时间维度上对系统多样性的拓展。

(5) 随机性。是指在系统中的构件执行过程中由指令集合、地址空间的变化所引入的不确定性。随机性方法在运行环境、软件层和数据层存在着广泛的应用。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)[16-17]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。将随机性分为静态随机性和动态随机性。若在系统运行前产生的不确定性并且在系统运行中不发生改变，称之为静态随机性；若在系统运行前后和系统的运行过程中都可能产生变化，则称之为动态随机性。静态随机性反映了系统构件间的差异程度所带来的不确定性，属于异构性范畴；动态随机性则反映了系统构件间的动态切换所带来的不确定性，属于动态性范畴。

1.2 DHR 架构

DHR 架构通过构造异构性和冗余性，并融入动态性、随机性和多样性，以对抗漏洞攻击。首先通过异构性的构造和组合方法，实现非相似余度架构(Dissimilar Redundancy Architecture，DRA)[2]，如图 1 所示。

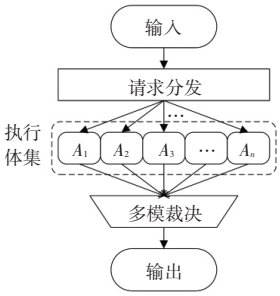


图 1 DRA 模型

Fig. 1 DRA model

由图 1 可知，DRA 由输入、请求分发器、异构执行体集合、多模表决器和输出组成。在保证执行体间异构性足够大的情况下，利用共有漏洞互不重合的性质，阻止入侵行为。利用 n 个执行体间的异构性给系统带来安全增益，同时利用冗余性进行多模表决来提高系统输出正确性。可以看出，异构性和冗余性是构成 DRA 的基本属性。

在 DRA 模型的基础上，以异构性最大化为基础，冗余裁决为核心，综合动态性、随机性和多样性实现了 DHRA，如图 2 所示。

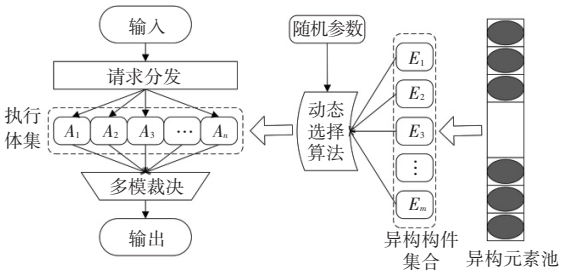


图 2　DHRA 模型

Fig. 2　DHRA model

由图 2 可知，DHRA 除了具备 DRA 的组成部分，增加了异构元素池、异构构件集合、动态选择器、随机发生器。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，通过建立异构元素池，从中选取异构元素，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 A，并按照这 n 个执行体的输出结果进行多模裁决，从而获得威胁感知能力。这里，设 P_A 表示执行体失效概率，V 表示裁决状态，则裁决后的失效概率 P_{judge} 为：

$$P_{judge} = \prod_{i=1}^{n} P_{A_i} V \quad \text{s.t.} \quad P_{A_iA_j} = P_{A_i}P_{A_j}, i \neq j \quad V = \begin{cases} 1 & \text{true} \\ 0 & \text{false} \end{cases} \tag{1}$$

理论上，DHRA 要求完全地去关联化和去协同化，以达到完全屏蔽互不相交的“有毒带菌”的异构构件漏洞后门威胁的目的。而在实际应用中，DHRA 的异构冗余构件非相似性设计则比经典的非相似余度要宽松得多。放宽对异构性设计的要求，一方面因为在实际的网络空间环境中，考虑到服务对象的功能、性能以及用户使用习惯等因素，人为设计存在处于同一概率空间的可能。也就是说，构造异构构件的过程之中，总会产生部分同构的成分。但是，由于有动态性的存在，只要保证在运行机制、调用机制、运行环境、参数赋值等方面存在不确定性，相互独立的构件之中的同构成分也就很难同时形成一致性或多数相同的错误。另一方面，由于在现有的软硬件体系结构中已经包含了一些安全防御手段，对于已经相对安全的系统来说，放宽相异性在一定程度上能够提高系统工作效率。

2　DHR Web 威胁感知系统

Web 服务作为网络空间基础设施服务中关键组成部分，发展迅速且规模庞大。Web 服务的多元化发展趋势也为 DHR 架构模型的应用提供了天然的优势。因此，通过构造异构的 Web 威胁感知容器，实现异构的 Web 服务软件栈，通过多模裁决技术实现基于 DHR 架构的 Web 威胁感知系统。

2.1　异构 Web 容器设计

在 DHR Web 威胁感知系统中，其安全性上限的高低往往依赖于各个 Web 服务执行构件之间的异构性大小。执行构件间的异构性越大，相交的漏洞威胁就越少，系统也就越安全。在异构 Web 服务容器中，异构性的实现方法主要包括：自然异构性、随机异构性和可控异构性。由此推得研究表述详见如下。

(1) 自然异构性。来自于软件开发过程，是在软件研发过程中，由于受到市场竞争、开发者、执行环境和开发语言等诸多社会因素的作用，自然而然形成的一种差异形式。目前，多元化的软件市场中充斥着大量的自然异构软件[13]。例如，操作系统软件，包括 Windows、Linux、Unix 为核心的自然异构操作系统；服务器软件，包括 Apache、Nginx、IIS、Tomcat、Lighttpd 等自然异构服务器；数据库软件，包括 Mysql、Sql Server、Oracle、PostgreSQL、Sybase、DB2 等自然异构数据库。

(2) 随机异构性。是指通过在应用代码自身或者代码执行过程中使用随机化方法产生差异的形式。随机异构性的产生通常应用在代码层面。静态的随机化可以在源代码层面或二进制代码层面产生相同程序的不同版本，典型的技术如代码混淆技术[18]。动态的随机化能够直接或间接地产生同一程序的不同执行过程集，是在执行环境中实现的，常用的技术如多样化编译等。

(3) 可控异构性。是通过人为控制来产生异构性，在保证程序功能等价的前提下通过人为构造多个独立开发的程序从而带来人为可控的差异。可控异构性常见于产品开发线、开源软件架构和多版本软件[19]。例如，操作系统、服务器、数据库等软件版本更替产生的异构性；开源软件的不同分支产生的异构性，等。

综上所述，对比 3 种异构性的实现方法可知，自然产生的异构性，在开发过程中通常会采用不同设计手段，因此其异构程度相对较大，但是利用自然异构的应用数量有限；随机方法产生的异构性，因为并不能对随机过程展开控制，因此无法预知其异构性表现；人为控制产生的异构性，通常会基于相同的构造方法实现进一步功能，因此其异构程度相对较小。

2.2 DHR Web 服务软件栈

根据 DHR 架构模型实现的 Web 服务采用层次化的设计方法，依据不同的功能定位选取相应的异构性实现方法，通过重构、重组等方法构成了异构的 Web 服务器软件栈。DHR Web 服务软件栈主要包括应用层、存储层、服务层、系统层等层次结构。如图 3 所示，在应用层使用诸如 Java、PHP 或 Python 之类的高级编程语言定制应用程序业务逻辑；在服务层接收来自客户端的 HTTP 请求，进行解析并将请求传递给相应的服务器端程序，如 Apache、IIS 和 Nginx 等；在存储层用来存储服务程序和用户数据，包括数据应用接口和存储的数据文件；在操作系统层为 Web 服务层和存储层提供运行时环境，如 Windows、Linux 和 Unix。

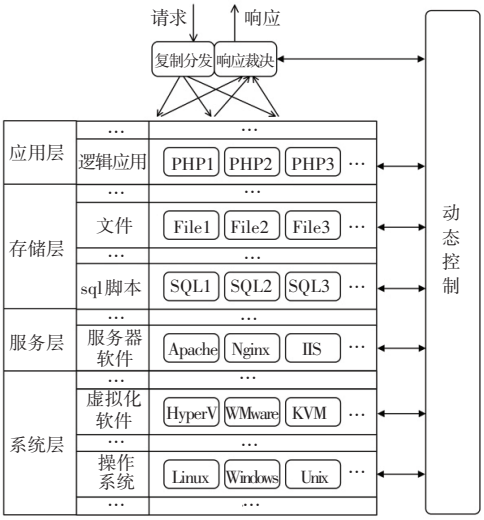


图 3 DHR Web 服务软件栈层次模型

Fig. 3 DHR Web service software stack layered model

在 DHR Web 服务的软件栈层次结构中，每一层都有可能遭受攻击威胁。例如，攻击者利用缓冲区溢出的攻击方法对系统层进行攻击，获取系统应用权限；或者攻击者通过利用应用层漏洞，在 Web 应用程序中上传并执行恶意脚本，获取用户信息，通过进一步提升权限，获得整个网站的管理特权；又或攻击者通过利用存储层漏洞，在 Web 应用程序中进行 SQL 注入攻击，获取存储数据，通过进一步提升权限，获得数据库管理权限。因此，在 DHR Web 服务软件栈中，通过使用异构 Web 容器设计方法，在系统的各个层次增加异构性成分，使得 Web 服务能够更好地应对复杂多样的攻击手段，以增加攻击者攻击的复杂性和成本，阻断攻击威胁。

基于此，将多个层次的异构组件进行动态组合，形成异构的 Web 服务软件栈 。对于不同层次组成元素失效率分别用 λ_{os}、λ_{ds}、λ_{ws}、λ_{al} 来表示；P_{os}、P_{ds}、P_{ws}、P_{al} 依次表示图 3 中操作系统层、假设失效在时间维度上可以看作是均匀的随机事件且各层次失效概率为独立事件，则有服务层、数据存储层、应用逻辑层的失效概率为：

$$\begin{cases} P_{os_i} = 1 - e^{-\lambda_{os_i} t} & P_{ds_i} = 1 - e^{-\lambda_{ds_i} t} \\ P_{ws_i} = 1 - e^{-\lambda_{ws_i} t} & P_{al_i} = 1 - e^{-\lambda_{al_i} t} \end{cases} \tag{2}$$

单个执行体的失效率 λ_i 为：

$$\lambda_i = \lambda_{os_i} + \lambda_{ds_i} + \lambda_{ws_i} + \lambda_{al_i} \quad i \in [1, n] \tag{3}$$

单个执行体的失效概率 F_i 为：

$$\begin{aligned} P_i &= P(\bar{M}_{os_i} \cup \bar{M}_{ds_i} \cup \bar{M}_{ws_i} \cup \bar{M}_{al_i}) = \\ &1 - (1 - P_{os_i})(1 - P_{ds_i})(1 - P_{ws_i})(1 - \\ &P_{al_i}) = 1 - e^{-(\lambda_{os_i} + \lambda_{ds_i} + \lambda_{ws_i} + \lambda_{al_i}) t} = \\ &1 - e^{-\lambda_i t} \end{aligned} \tag{4}$$

2.3 基于裁决的威胁感知方法

在 Web 威胁感知系统中 Web 威胁感知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，比较异构的 Web 威胁感知容器的响应输出结果，可得相对正确的响应输出结果，从而达到屏蔽漏洞后门和感知攻击威胁的目的。基于裁决的 Web 威胁感知方法更多地被视为一种响应比较的策略应用于 DHR 架构中。具体到不同的应用场景，实现方法会有一定的差异，常采用的裁决实现方法包括全体一致裁决、大数裁决算法、最大近似裁决、基于历史信息的带权裁决等。基于裁决的 Web 威胁感知方法如图 4 所示。

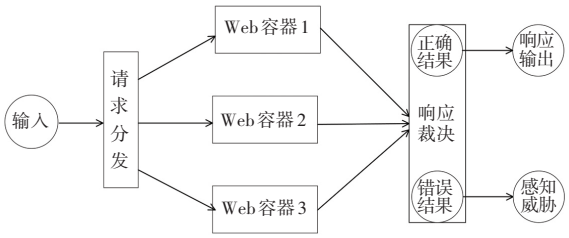


图 4 基于裁决的威胁感知方法

Fig. 4 Threat awareness based on adjudication

基于裁决的 Web 威胁感知方法，在 Web 服务入口和出口处，增设了请求分发和响应比较模块作为输入和输出代理。当请求到来时，分发器将请求复制为多份，分发至多个异构的 Web 容器中，响应裁决模块通过收集比较多个响应的异同，得到唯一的响应输出，同时判断异常信息记录到威胁感知日志当中，以此来获得安全增益。

在不同的 DHR 架构下，Web 威胁感知方法的安全

增益是不同的。对 k/n 架构下的威胁感知方法进行安全增益，假设对系统中不同执行体的失效率是相同的，即 $P_i = P_j = P$。且 2 个执行体间的失效是相互独立的，即 $P_{ij} = P_i \cdot P_j = P^2$，$(i \neq j)$。则威胁感知系统失效概率即为至少有 k 执行体失效的概率，数学表述如下：

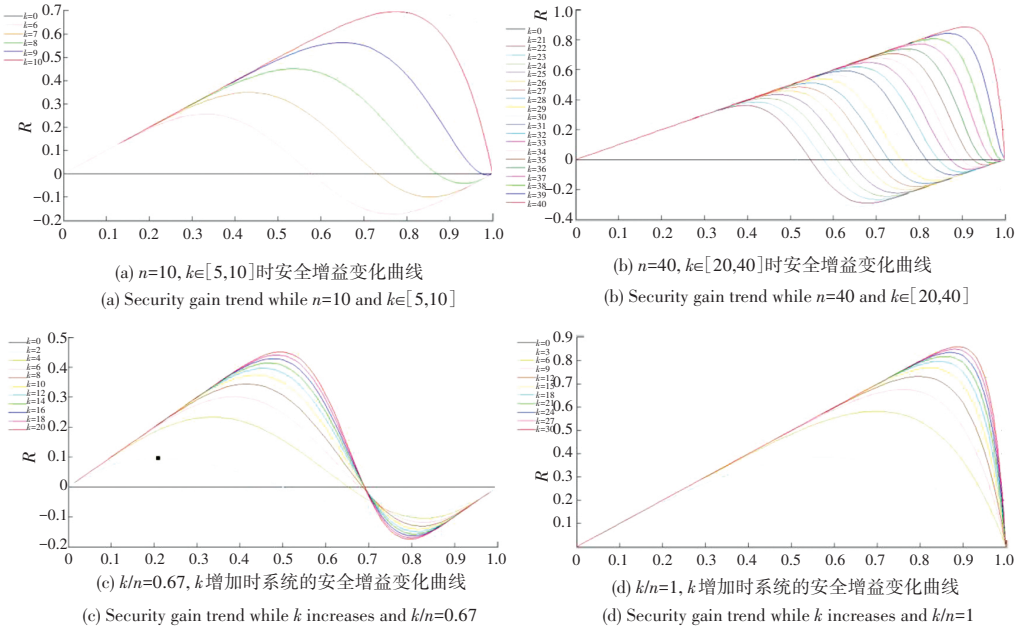
$$F_{mimic} = \sum_{i=k}^{n} M(i) = \sum_{i=k}^{n} \binom{i}{n} F^i \cdot (1-F)^{n-i} \quad (5)$$

于是有系统安全增益为：

$$\Delta_{security} = P - \sum_{i=k}^{n} \binom{i}{n} P^i \cdot (1-P)^{n-i} \quad (6)$$

3　实验分析

根据 Web 威胁感知系统的安全增益计算方法，讨论在不同的 DHR 架构下安全增益的变化趋势，如图 5 所示。在图 5 中，图 5(a) 的趋势图为：当 $n = 10$ 固定不变时，k 取 $n/2$ 到 n 的安全增益变化曲线；图 5(b) 的趋势图为：当 $n = 40$ 固定不变时，k 取 $n/2$ 到 n 的安全增益变化曲线；图 5(c) 的趋势图为：当 $k/n = 0.67$ 固定不变、k 增加时，系统的安全增益变化曲线；图 5(d) 的趋势图为：当 $k/n = 1$ 固定不变、k 增加时，系统的安全增益变化曲线。



(a) n=10, k∈[5,10]时安全增益变化曲线
(a) Security gain trend while n=10 and k∈[5,10]

(b) n=40, k∈[20,40]时安全增益变化曲线
(b) Security gain trend while n=40 and k∈[20,40]

(c) k/n=0.67, k增加时系统的安全增益变化曲线
(c) Security gain trend while k increases and k/n=0.67

(d) k/n=1, k增加时系统的安全增益变化曲线
(d) Security gain trend while k increases and k/n=1

图 5　安全增益变化趋势图

Fig. 5　Security gain trend

通过对图 5 进行对比分析可知，当 n 不变时，k 越大则安全增益为正的失效概率范围越大，安全增益的最值越高；当 n 增大时，安全增益为正的失效概率范围不一定增大，但是安全增益的最值会提高；当 k/n 值不变时，安全增益为正的失效概率范围基本保持不变，安全增益最值会提高；当 k/n 值不变时，安全增益为正的失效概率范围基本会随之增大，安全增益最值会提高。这说明了不同的 DHR 架构下，冗余程度 n 只会对威胁感知的上限产生影响，而裁决精度 k/n 则会对威胁感知容器的选取产生影响。

4　结束语

本文从 DHR 架构的防御特性和架构模型出发，阐述了 DHR Web 威胁感知的设计方法。在不同 DHR 架构下，分析 Web 威胁感知系统的安全增益，为 DHR Web 威胁感知技术的进一步研究和发展提供依据和参考。在未来，如何通过细粒度的异构性构造方法构造 Web 威胁感知容器，从而最大化提升 DHR 架构下的 Web 威胁感知系统的安全增益，将会成为 DHR Web 威胁感知技术发展的方向。

参考文献

[1] CHERDANTSEVA Y, BURNAP P, BLYTH A, et al. A review of cyber security risk assessment methods for SCADA systems [J]. Computers & Security, 2016, 56(C): 1-27.

[2] 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.

[3] 邬江兴. 网络空间拟态安全防御 [J]. 保密科学技术, 2014(10): 4-9.

[4] 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.

[5] 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.

(下转第 47 页)

特征后，PCA、ADTree、One－class SVM、Naive Bayes 这 4 种算法的检测效率相比没有使用新特征全部得到了提高，4 种算法的误检率相比没有使用新特征全部得到了降低。其中，Naive Bayes 的准确率提升的幅度最大，提升了 9.4 个百分点。One－class SVM 的误检率降低的幅度最大，降低了 2.7 个百分点。

5　结束语

通过对大量恶意代码的研究，对混淆恶意 JavaScript 代码进行四大特征分类，并且基于四大特征类中的新、旧特征展开了研究与检测分析，获得了满意的结果。过程中研究了大量混淆代码，试图提取出尽可能多的静态混淆特征，但是攻击者仍然在继续开发着不同的混淆技术，致力于掩盖 JavaScript 的真实结构。因此在某些情况下，某些被混淆的恶意 JavaScript 代码是无法被检测到的。此后的下一步工作是继续深入研究，找出共性，还原其本质，通过提取新特征以及改进检测算法来提高运行结果的检测效率，在攻与防的对立统一中，寻求重大的技术突破。

参考文献

[1] 马洪亮，王伟，韩臻. 混淆恶意 JavaScript 代码的检测与反混淆方法研究[J]. 计算机学报，2017，40(7)：1699－1713.

[2] WANG Weihong，LV Yinjun，CHEN Huibing，et al. A static malicious JavaScript detection using SVM [C]//Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering. Paris，France：Atlantis Press，2013：214－217.

[3] CURTSINGER C，LIVSHITS B，ZORN B G，et al. ZOZZLE：Fast and precise in－browser JavaScript malware detection [C]// 20th Usenix Security Symposium. San Francisco，CA：Usenix，2011：33－48.

[4] KOLTER J Z，MALOOF M A. Learning to detect and classify malicious executables in the wild [J]. Journal of Machine Learning Research，2006，7：2721－2744.

[5] RIECK K，KRUEGER T，DEWALD A. Cujo：Efficient detection and prevention of drive－by－download attacks [C]//Proceedings of the 26th Annual Computer Security Applications Conference. Austin，Texas，USA：ACM，2010：31－39.

[6] COVA M，KRUEGEL C，VIGNA G. Detection and analysis of drive－by－download attacks and malicious JavaScript code [C]// Proceedings of the 19th International Conference on World Wide Web. Raleigh，North Carolina：ACM，2010：281－290.

[7] 徐青，朱焱，唐寿洪，等. 分析多类特征和欺诈技术检测 JavaScript 恶意代码[J]. 计算机应用与软件，2015，32(7)：293－296.

[8] PATIL D R，PATIL J B. Detection of malicious JavaScript code in Web pages [J]. Indian Journal of Science and Technology，2017，10(19)：1－12.

[9] FERNANDEZ F. Heuristic engines [C]//Proceedings of the 20th Virus Bulletin Conference. Vancouver：[s.n.]，2011：407－444.

[10] SCHÖLKOPF B，PLATT J C，SHAWE－TAYLOR J，et al. Estimating the support of a high－dimensional distribution [J]. Neural computation，2001，13(7)：1443－1471.

(上接第 41 页)

[6] HAN Jin，ZANG Binyu. Analyzing the effectiveness of software diversity for system security [J]. Computer Applications & Software，2010，27(9)：273－275，300.

[7] 张宇嘉，庞建民，张铮，等. 基于软件多样化的拟态安全防御策略[J]. 计算机科学，2018，45(2)：215－221.

[8] LI Weichao，ZHANG Zheng，WANG Liqun. Improvement in diversify active defense for web application by using language and database heterogeneity [C]// International Conference on Anti－Counterfeiting，Security and Identification. Xiamen，China：IEEE，2017：30－35.

[9] 马博林，张铮，刘健雄. 应用于动态异构 web 服务器的相似度求解方法[J]. 计算机工程与设计，2018，39(1)：282－287.

[10] MA Bolin，ZHANG Zheng，ZHU Yongsheng. A formalization research on web server and scheduling strategy for heterogeneity [C]// Advanced Information Management，Communicates，Electronic and Automation Control Conference. Xi´an，China：IEEE，2016：1447－1451.

[11] BIRMAN K，SCHNEIDER F B. Security risks from a software monoculture [R]. New York：Cornell University，2008.

[12] HERSCHEL I F W. Report on Mr. Babbage´s calculating engine [J]. Journal of the Franklin Institute，1831，11(3)：210－213.

[13] BAUDRY B，MONPERRUS M. The multiple facets of software diversity [J]. Acm Computing Surveys，2015，48(1)：1－26.

[14] PENG Wei，LI Feng，HUANG C T，et al. A moving－target defense strategy for cloud－based services with heterogeneous and dynamic attack surfaces [C]// IEEE International Conference on Communications. Sydney，NSW，Australia：IEEE，2014：804－809.

[15] LEW H L，DIKMEN S，Slimp J，et al. Organically assured and survivable information systems (OASIS) demonstration and validation program [J]. American Journal of Physical Medicine & Rehabilitation，2003，82(1)：53－61.

[16] VOAS J，GHOSH A，CHARRON F，et al. Reducing uncertainty about common－mode failures [C]// Eighth International Symposium on Software Reliability Engineering. Albuquerque，NM：IEEE Computer Society，1997：308.

[17] NADUNGODAGE C H，XIA Y，VAIDYA P S，et al. Online multi－dimensional regression analysis on concept－drifting data streams [J]. International Journal of Data Mining Modelling & Management，2014，6(3)：217.

[18] VOLCKAERT S，COPPENS B，De SUTTER B. Cloning your gadgets：Complete ROP attack immunity with multi－variant execution [J]. IEEE Transactions on Dependable & Secure Computing，2016，13(4)：437－450.

[19] GRUZENKIN D V，CHERNIGOVSKIY A S，TSAREV R Y. N－version software module requirements to grant the software execution fault－tolerance [C]// Computational Methods in Systems and Software. Cham：Springer，2017：293－303.