计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(2):240−245 [doi: 10.15888/j.cnki.csa.006721]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于随机森林的 WebShell 检测方法①

秦　英

(武汉邮电科学研究院, 武汉 430074)
(南京烽火星空通信发展有限公司, 南京 210019)
通讯作者: 秦　英, E-mail: Qying3375@163.com

摘　要: WebShell 根据其功能和大小可以分为多种类型, 各种类型的 WebShell 在基本特征上又有其独有的特征, 而现有的 WebShell 检测大多从单一层面提取特征, 无法较全面的覆盖各种类型 WebShell 全部特征, 具有种类偏向性, 无差别的检测效果差, 泛化能力弱等问题. 针对这一问题, 提出了一种基于随机森林的 WebShell 检测方法. 该方法在数据预处理阶段分别提取文本层的统计特征和文本层源码与编译结果层字节码 (opcode) 的序列特征, 构成较全面的组合特征, 然后通过 Fisher 特征选择选取适当比例的重要特征, 降低特征维度, 构成样本的特征集, 最后采用随机森林分类器训练样本得到检测模型.通过实验表明, 本检测方法能有效地检测 WebShell, 并在准确率、召回率和误报率上都优于单一层面的 WebShell 检测模型.

关键词: WebShell; 随机森林; 组合特征; 特征选择

引用格式: 秦英.基于随机森林的 WebShell 检测方法.计算机系统应用,2019,28(2):240–245. http://www.c-s-a.org.cn/1003-3254/6721.html

Webshell Detection Method Based on Random Forest

QIN Ying

(Wuhan Research Institute of Posts and Telecommunications, Wuhan 430074, China)
(FiberHome Communications Science &Technology Development Co. Ltd., Nanjing 210019, China)

Abstract: WebShell can be divided into various types according to its function and size; they have basic features and unique features. However, most existing WebShell detection only extracts features from single level, they cannot cover all the features of various types of WebShell in a more comprehensive way. These detections have problems such as kind bias, poor detection effect, weak generalization ability, etc. To solve these problems, a random forest based WenShell detection method is proposed. In the data preprocessing stage, this method extracts the statistical features of the text layer, and the sequence characteristics of the text layer sources and the compilation result layer opcode, to form a comprehensive combination features. Then, the feature set of the sample is formed by using Fisher feature selection to select important features with the appropriate proportion to reduce the feature dimension. Finally, the random forest classifier is used to train samples to get the detection model. The experiment shows that this detection method can detect WebShell effectively, and it is superior to the single level WebShell detection model in accuracy, recall, and false alarm rate.

Key words: WebShell; random forest; combination features; feature selection

随着 Internet 技术的兴起和 WWW 浏览器技术的不断成熟, 人们已经熟悉浏览器的上网方式, 使得基于 B/S 结构的 Web 应用在信息系统的开发建设中盛行起来.但由于 Web 系统开发人员的技术水平参差不齐, 安全意识淡薄, 防护能力不足, Web 系统不断面临网络安全威胁.例如, Web 交互功能越来越丰富, 人们可以快

① 收稿时间: 2018-06-29; 修改时间: 2018-07-20; 采用时间: 2018-08-08; csa 在线出版时间: 2019-01-28

速便捷地传递信息、分享与获取资源, 但这也为黑客提供了多方面的攻击手段.根据 CNCERT 今年 4 月发布的《2017 年我国互联网网络安全态势综述》显示, CNCERT 监测发现境内外约 2.4 万个 IP 地址对我国境内 2.9 万余个网站植入后门. 网站后门又被称为 WebShell, WebShell 危害巨大, 如果发现网站被植入了 WebShell 脚本, 则意味着网站存在漏洞, 而且攻击者已经利用漏洞获得了服务器的控制权限[1]. 因此检测 WebShell 对于及时掌握网络安全态势具有极其重要的意义.

1 相关研究

1.1 WebShell 概述

目 WebShell 是运用主流脚本语言如 PHP、JSP、ASP 等编写的一种网页脚本木马. 攻击者在检测到 Web 应用存在上传漏洞后, 常常将这些脚本木马放置在网站服务器的 Web 目录中, 然后以访问网页的方式访问脚本木马, 通过脚本木马获取更高的权限以控制网站服务器, 对网站服务器实施文件的上传、修改与下载、访问数据库篡改数据、执行任意程序命令等恶意操作[2].

WebShell 具备很深的自身隐藏性、可伪装性, 它混杂在正常网页中, 和正常网页一样通过 80 端口与服务器或远程主机交换机进行数据传递. 这种数据传递属于正常的 HTTP 协议, 传统防火墙无法对其进行拦截, 也不会在系统日志中留下服务器管理操作, 这给 WebShell 的检测带来很大的难度.

1.1.1 WebShell 的特征

根据脚本程序的大小和功能, 攻击者通常将 WebShell 分为大马、小马和一句话木马三类[1].

不同类型的 WebShell 不仅具有基本的 WebShell 特征, 又有各自类型的独有特征[3]. WebShell 的基本特征有: (1) 访问特性: WebShell 的访问出度和入度、访问 IP 较正常网页访问有很大区别; (2) 关联特性: WebShell 一般是个孤立的网页文件, 在页面跳转上存在特殊性; (3) 文件特性: WebShell 在文件创建时间、文件行数、文件大小上一般具有特殊性[1]. 在文件大小和文件行数上明显区别于一般合法网页; (4) 文本特性: 包含敏感函数的调用, 或是通过加解密算法 (混淆处理) 隐藏敏感函数调用.

各类 WebShell 独有特征:(1) 大马的文件大小超过 70% 的合法网页、调用多种系统关键函数、包含开发者的版权信息, 如 4ngel、wefoiwe、c99shell 等; (2) 小马文件小, 功能单一, 存在上传行为; (3) 一句话木马长度短、可能嵌入到正常网页中、通过 POST 方式传递参数.

1.2 WebShell 的检测和逃逸

WebShell 检测和逃逸如同一场旷日持久的博弈, 脚本语言的灵活多变使得 WebShell 不断变异躲避检测[4], 而检测也从不同角度查杀 WebShell.

1.2.1 WebShell 逃逸技术

以 PHP 语言的 WebShell 为例, 目前常见的 WebShell 变形技术有:

(1) 加密算法隐藏敏感特征: 加密的方式非常多, 在 PHP 语言中最常用的是内置的 base64, rot13 等加密技术, 研究 WebShell 样本可以发现很多的 WebShell 其实是 eaval (base64_decode($_POST[X])) 的变形. 为了逃逸, 有些恶意程序会自定义加密算法, 或使用 Weevely、phpjm、phpjiami 等加密工具加密混淆代码, 隐藏特征函数.

(2) 字符串拆分变形重组技术: 在字符串中插入注释, 字符替换, 或是通过一些特殊字符的异或运算得到特定字符串, 如“$__=("#"^"|").("."^"~").("/"^"‘").("|"^"/").("{"^"/")”, “$__”结果为“_POST”. 字母多次自增或自减可以得到任意字母, 再利用字符串的拼接特性同样也能得到特定函数, 比如“b”可由“a++”表示.

(3) 隐蔽位置传递参数: 检测未被查杀软件扫描的字段, 如“User-Agent”、“Accept-Language”等, 利用这些字段传递参数. 一些文件的描述信息也常会放置 WebShell 中需要的数据, 如图片的 exif 信息. 此外, 还可以将敏感函数名作为文件名的一部分, 再在文件中访问文件名得到敏感函数.

1.2.2 现有 WebShell 检测技术

目前 WebShell 的检测, 可以大体分为脚本运行前的静态检测, 脚本运行后的日志检测, 和脚本运行中的动态检测.

静态检测: 传统的静态检测是基于特征库的匹配, 如 WebShell Detector 有一个较大的特征库, 知名 WebShell 检测器 D 盾对静态属性进行匹配. 这类基于特征字符的匹配一般通过正则表达式来实现, 而正则表达式具有无法完整覆盖风险模型的问题[5], 会造成一定的漏报和误报, 而且攻击者使用混淆手段很容易躲

避这类检测. 基于统计学的检测, 如开源项目 NeoPI[6]提取文件的信息熵、最长单词、重合指数和压缩比检测混淆 WebShell, 胡建康[7]提取内容属性、基本属性和高级属性作为决策树 C4.5 的分类特征. 这类方法在混淆 WebShell 检测上有一定的效果, 但是近年随着人们产权和安全保密意识的加强, 许多开发者也会使用加密技术隐藏自己的代码逻辑, 从而呈现出类似的统计特征, 这样便增大了统计检测误报的可能性. 静态检测不只停留在源码层面, 还从词法分析语法角度、语义角度、字节码角度进行恶意代码检测, 减小代码注释和混淆变形对检测结果的影响.

动态检测: 传统的动态检测主要为分析脚本执行过程中的动态特性检测恶意脚本, 包括基于行为的 WebShell 检测和基于流量的 WebShell 检测. 基于行为的 WebShell 检测, 以时间为索引, 从行为模式上分析脚本对文件的、对系统的操作来判断是否为 WebShell[8]. 基于流量的检测方法, 在 HTTP 请求/响应中寻找可疑信息, 分析访问的访问特征 (IP/UA/Cookie), path 特征, payload 的行为特征区分 WebShell 与合法网页. 动态的检测方式, 不仅能有效检测出已知的 WebShell, 还对未知的、伪装性强的 WebShell 有较高的检出率, 但只能检测到正在发生上传或访问 WebShell 的行为, 对于网站中已有的且未使用 WebShell 无法检测, 存在工作量大, 误报率高等问题, 在检测效果上没有特征值检测效果好, 还需深入研究.

日志检测: 虽然 WebShell 与合法网页一样通过 80 端口进行 HTTP 请求/响应, 不会在系统日志中留下记录, 但分析 Web 服务器日志时, 可以在页面访问频率、页面关联度、访问 IP 等日志文本特征中找到 WebShell 的蛛丝马迹. 分析完整的日志上下文, 能获取黑客的攻击意图和还原整个攻击过程, 有助于发现未知、变种的 WebShell 脚本. 但其缺点也很明显, 日志分析技术通常需要通过大量的日志数据来建立 HTTP 请求异常模型, 数据处理起来耗时长、检测速度慢, 会影响服务器的性能. 而且 Web 服务器业务功能复杂, 存在一定数量的网页很少被外界访问, 使得访问次数/频率不能作为明显特征, 另外, 部分 WebShell 数据访问时会模拟正常数据库操作, 这将对基于数据库访问规则的检测造成干扰从而导致误报.

综上所述, 基于各种角度的 WebShell 检测有各自的优点, 也存在各自的不足. WebShell 的检测困难主要来自如下方面: (1) 无差别的检测: 虽然机器学习已经应用到了 WebShell 检测中, 但是可以发现不同类型的 WebShell 依赖特征不同, 特征覆盖不全使得漏报误报问题明显, 无差别的检测有待提高. (2) 特征不突出: 随着人们对产权和安全保密意识的加强, 许多开发者也会使用加密技术隐藏自己的代码逻辑, 呈现出类似 WebShell 的特征, 增大了检测误报的可能性.

2 基于随机森林的 WebShell 检测方法

要解决以上问题, 特征的提取和分类算法的选择至关重要. 现有的基于机器学习算法的 WebShell 检测模型大多从单一层面提取特征, 无法覆盖各种类型 WebShell 的全部特征, 具有种类偏向性, 无差别的检测效果差, 泛化能力弱等问题. 对于以上不足, 本文提出一种新的基于随机森林的检测模型 (其框架如图 1 所示), 创新性地将页面文本层的特征和 PHP 编译结果层的特征结合, 建立组合的 WebShell 特征集, 同时通过特征选择方法解决特征维度过大导致特征冗余、模型过拟合的问题. 在分类器上, 本文利用组合分类器算法—随机森林, 进行模型训练和分类, 利用随机森林的强大的泛化能力, 提升模型检测准确率.

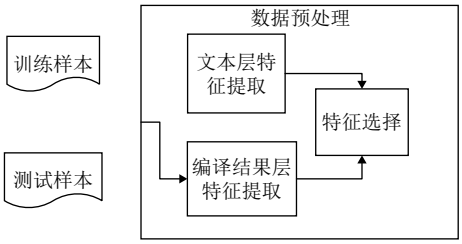

图 1 检测模型框架

2.1 数据预处

由于 PHP, ASP, JSP 等不同语言编写的 WebShell 非常相似, 且在已知的 WebShell 中 PHP 编写的 WebShell 占得比重较大, 因此本文主要分析用 PHP 编写的 WebShell 的数据处理方法. 对其提取文本层和编译结果层的组合特征, 再进行特征选择, 建立降低维度后的数据特征集.

2.1.1 特征提取

(1) 文本层静态统计特征:

本文参考 NeoPI 的检测特征, 并加入非字母数字字符占比和基于特征码的匹配结果. 经过字符运算转

换的混淆 WebShell 页面其非字母数字字符占比一般大于正常页面. 同时传统的 WebShell 检测是基于特征码匹配的, 加入基于特征码的匹配结果能检测出已知的非正常函数调用, 如: 关键函数、系统函数、数据库和加解密函数的调用情况. 具体的统计特征提取方法如表 1 所示.

表 1 统计特征提取方法

统计特征	方法/公式	说明
最长字符串长度	提取出最长连续字符串	记录长度, 一般合法页面小于 150
文件重合指数 IC	$IC=\frac{\sum_{i=o}^{255} f_i(f_i-1)}{n(n-1)}$	i 为字符对应的 $ascii$ 值, n 为所有字符个数, 为字符 i 出现的频数
信息熵	$H=-\sum_{x\in\chi} p(x)\log(p(x))$	χ 为样本的所有字符, $p(x)$ 表示字符 x 的概率函数
文件压缩比	未被压缩的文件大小和压缩后的文件大小之比	经过编码混淆的文件其压缩比会变大
非字母数字字符占比	特殊字符总数和文件字符总数之比	经过字符转换、加密混淆的文件非字母数字字符比重比正常文件大
特征码匹配	正则匹配	样本匹配特征码文件中的规则标记为 1, 否则标记为 0

(2) 利用 N 元语言模型进行特征提取

1) 文本层序列特征

WebShell 的种类和变形逃逸手段虽然多样, 但都离不开 shell 的基本结构, 即数据的传递和执行传递的数据, 逃逸手段同样围绕传递参数的隐藏和执行方法的变形展开. 目前我们没有非常完善的方法定位数据传递的步骤, 但可以比较方便的找到数据执行的位置: “(”(使用小括号并不是代码执行的唯一方法). 本文在代码文本层采用一元语法模型 (1-gram) 切分样本, 切分出以左小括号结尾的连续字符串和以及字符串常量, 分别作为一个词组, 同时统计每个词组在样本中出现的次数, 得到文本层基于函数和字符串常量的词频矩阵, 将其作为特征向量.

2) opcode 序列特征

opcode 是 PHP 编译结果层的字节码, 是脚本编译后的中间语言. PHP 脚本在读入脚本程序字符串后, 经由词法分析器将其转换为语言片段 Tokens, 接着由语法分析器从中发现语法结构生成抽象语法树 AST, 再经过静态编译器便生成了对应的 opcode. 最后由 Zend 虚拟机执行每一条 opcode 得到运行结果. 通过 PHP 的扩展程序 VLD 可以直接得到脚本对应的 opcode 文本.

本文采用二元语法模型 (2-gram) 切分 opcode 文本, 将相邻的两个 opcode 划分到一个词组中, 统计该词组在每个样本中出现的次数[9], 得到编译结果层基于 opcode 序列的词频矩阵, 将其作为特征向量.

2.1.2 特征选择

应用上节的特征提取方法可以得到结合代码文本层和编译结果层的特征集, 但是其维度可能非常高, 会为后续模型训练带来巨大的计算压力, 本文在数据预处理阶段采用特征选择方法, 以降低特征维度, 降低后期模型训练复杂度, 节省数据存储空间. 特征选择是机器学习、模式识别领域的研究热点之一, 通过从原始的高维特征中筛选出符合要求的特征子集以达到降低特征空间的目的[10]. 特征选择主要有过滤式和封装式两种框架, 也有嵌入式和混合式方法. Fisher 线性判别是基于距离度量的过滤式特征选择之一, 比基于互信息度量的特征选择方法计算量小、准确率高, 可操作性强、节省运算时间, 对高维数据在维数约简与分类性能上有很好的效果. 因此本文使用 Fisher 线性判别算法进行特征选择, 降低特征维度.

Fisher 线性判别的思想是: 寻找一个合适的投影轴, 使得样本投影到这个轴上时同一类的投影点应可能靠近, 不同类的投影点尽可能远离, 即类内离散度尽可能小, 类间离散度尽可能大. 线性判别分析时利用了样本的类别信息, 因此是有监督的方法, 其目标函数表示为:

$$J(w)=\frac{w^{\mathrm{T}}S_b w}{w^{\mathrm{T}}S_w w} \tag{1}$$

其中, w 为投影方向, S_b 类间散布矩阵, S_w 为类内散布矩阵, J 的值越大, w 为的判别能力越强. Fisher 特征选择以特征作为投影轴, 计算该特征方向的判别值, 将每个特征根据其判别值从大到小排序, 判别值越大, 说明该特征对于分类的有效性越高, 该特征也越重要. 本文将特征提取后的各个特征根据其 Fisher 判别值进行从大到小排序, 按适当的比例选择出重要特征构成新的特征集, 用于之后的模型训练.

2.2 模型训练

完成对样本的特征提取后, 即可将特征矩阵作为

输入, 将标注结果作为预期输出, 训练分类器. 在分类器上, 本文选择随机森林算法对样本特征数据进行学习. 随机森林是集成学习器的一种, 具有分析复杂相互作用特征的能力, 对于噪声数据和存在缺失值的数据具有很好的鲁棒性, 并且具有较快的学习速度[11], 被誉为“代表集成学习技术水平的方法”.

随机森林模型由多棵 CART 决策树组成, 每一棵 CART 决策树都是通过两个随机过程进行构建的, 其具体生成步骤如下:

(1) 按照 Bagging 算法随机从训练样本数据集中有放回地抽取 K 个训练样本, 用于构建 K 棵决策树;

(2) 设样本有 n 个特征, 从每个训练样本的特征集中随机抽样 m 个特征作为每棵决策树的新的特征集 ($m \leqslant n$);

(3) 利用随机抽取获得的训练样本集和子特征集进行 CART 决策树模型的构建, CART 决策树根据基尼指数来选择划分属性, 进行节点分裂. 每棵树最大限度地生长, 不做任何剪枝, 形成随机森林.

(4) 数据预测时, 利用所有的决策树进行预测, 最后以投票的方式决定模型最终的预测结果.

3 实验

3.1 实验数据

本文将 PHP 语言编写的 WebShell 脚本作为研究对象, 共收集了 1000 个恶意 PHP WebShell 样本和来自 PHPCMS、WordPress 等开源项目的 2000 个正常 PHP 样本作为实验数据. 实验将样本随机分为 4 份, 任选 3 份训练模型, 剩下的样本用于测试模型的检测能力, 最终结果为运算 10 次后的平均结果.

3.2 评估准则

在本实验中, WebShell 标记为 1, 正常样本标记为 0, 分类结果混淆矩阵如表 2 所示.

表 2 分类结果混淆矩阵

真实情况	预测结果	
	1	0
1	*TP*(正真例)	*FN*(假反例)
0	*FP*(假正例)	*TN*(真反例)

为了评估实验效果, 本文采用三个指标评估检测方法性能: 准确率、召回率和误报率. 准确率 (acc) 定义为 $\frac{TP+TN}{TP+FN+FP+TN}$, 召回率 (recall) 定义为 $\frac{TP}{TP+FN}$, 误报率 (error) 定义为 $\frac{FP}{FP+TN}$.

3.3 实验分析

实验分三步进行: (1) 将特征提取后的各个特征根据其 Fisher 判别值进行从大到小排序, 分析选择不同比例的重要特征集对随机森林模型检测效果的影响, 从中选取适当比例的特征构建新的特征集, 其中随机森林决策树的个数为 10; (2) 根据选取的新特征集, 对比决策树数目对随机森林模型检测效果的影响, 从中确定取得较好平衡的决策树个数 treeNum 用于完成本文检测模型的构建; (3) 比较基于 SVM 的检测模型与本文构建的检测模型的检测效果.

图 2 反映了选择不同比例的特征集对随机森林模型检测效果的影响, 分析图 2 可得, 在特征根据 Fisher 判别值排序后, 选择前 10% 特征集训练的模型的准确率可达到 97% 左右, 说明 Fisher 特征选择可以选择出重要的特征, 降低特征维度. 当特征选择从 10% 增加到 30% 时, 检测准确率和召回率逐步上升, 误报率降低且稳定在 0.6% 左右. 而此后随着特征增加, 模型检测准确率、召回率和误报率都没有明显改善, 说明存在较多的冗余特征, 因此本文选择前 30% 的重要特征作为新的特征集.

在特征集选定后进行第二步实验, 由图 3 可以得出, 决策树数目 *treeNum* 从 1 增加到 50 时其检测性能也随之逐步提高, 当继续增加 *treeNum* 时性能提升缓慢, 在 *treeNum*=200 时准确率最高, 之后继续增加 *treeNum* 性能反而开始下降, 但训练时间却一直不断上升. 综合分析, 在决策树数目 *treeNum*=200, 特征集选取组合特征集的前 30% 的重要特征时, 本论文提出的检测模型达到最好效果.

表 3 对比了 3 种类型的 WebShell 检测模型: 文本层基于 SVM 和静态统计特征的 WebShell 检测模型、编译结果层基于 SVM 的检测模型和本文的组合层面基于 Fisher 特征选择和随机森林的检测模型. 可以得出 3 种检测模型都可以在较短的时间内完成检测任务, 综合对比, 编译结果层的检测效果优于文本层的基于静态统计特征的检测效果, 而本文提出的检测模型在准确率、召回率和误报率上都优于以上两种在单一层面的检测模型, 检测时间也小于 0.2 秒, 表明组合层面基于 Fisher 特征选择和随机森林的检测模型具有更好的拟合能力和更好的对未知样本的检测能力.

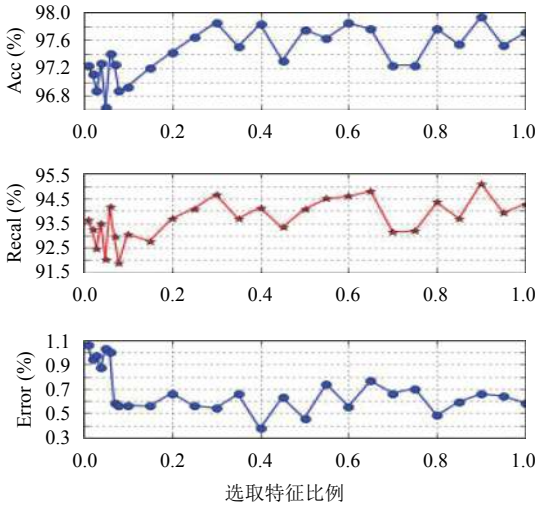

图 2 选取不同比例特征的检测效果

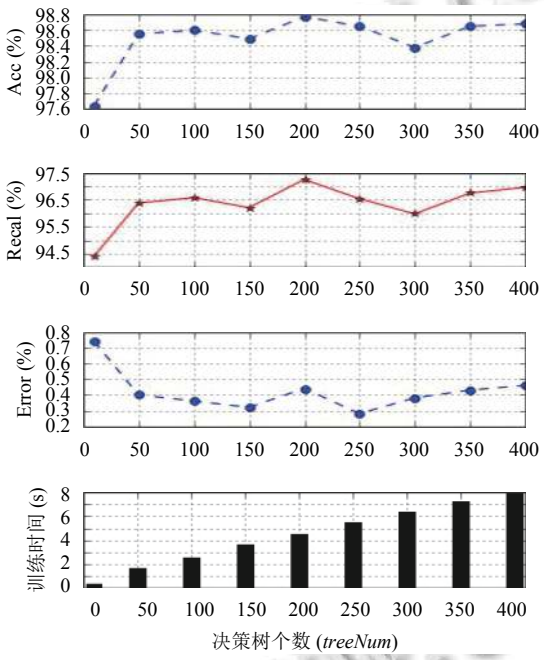

图 3 随机森林检测效果对比

表 3 各种模型检测效果

性能	文本层基于 SVM 和静态统计特征的检测模型	编译结果层基于 SVM 的检测模型	本文提出的检测模型
准确率 (%)	79.28	94.25	98.69
召回率 (%)	51.08	95.84	96.58
误报率 (%)	9.45	7.41	0.26
检测时间 (s)	0.037	0.008	0.128

4 结束语

本文深入分析了 WebShell 的基本特征和各种类型的独有特征, 以及 WebShell 逃逸技术, 指出现有 WebShell 检测方法从单一层面提取特征, 无法覆盖各种类型 WebShell 的全部特征, 无差别的检测效果差, 泛化能力弱的问题. 为解决以上问题, 本文提出了基于随机森林和组合特征的 WebShell 检测方法. 分别从文本层和编译结果层提取特征构建组合特征集, 并引入 Fisher 特征选择, 解决特征集过大特征冗余的问题. 在分类器上采用随机森林算法用以提高检测模型的泛化能力. 经过实验证明, 本文提出的方法具有很好的 WebShell 检测效果. 但是如果将该检测方法应用到实际的工程检测引擎中, 其检测效率和效果还得进一步探讨.

参考文献

1 叶飞, 龚俭, 杨望. 基于支持向量机的 Webshell 黑盒检测. 南京航空航天大学学报, 2015, 47(6): 924–930.

2 张红瑞. WebShell 原理分析与防范实践. 现代企业教育, 2013, (20): 254–255. [doi: 10.3969/j.issn.1008-1496.2013.20.218]

3 贾文超, 戚兰兰, 施凡, 等. 采用随机森林改进算法的 Webshell 检测方法. 计算机应用研究, 2018, 35(5): 1558–1561. [doi: 10.3969/j.issn.1001-3695.2018.05.060]

4 Mingkun X, Xi C, Yan H. Design of software to search ASP web shell. Procedia Engineering, 2012, 29: 123–127. [doi: 10.1016/j.proeng.2011.12.680]

5 Hansen RJ, Patterson ML. Guns and butter: Towards formal axioms of input validation. Black Hat USA, 2005, (8): 1–6.

6 Tu TD, Cheng G, Guo XJ, *et al*. Webshell detection techniques in web applications. Proceedings of the 5th International Conference on Computing, Communications and Networking Technologies (ICCCNT). Hefei, China. 2014. 1–7.

7 胡建康, 徐震, 马多贺, 等. 基于决策树的 WebShell 检测方法研究. 网络新媒体技术, 2012, 1(6): 15–19. [doi: 10.3969/j.issn.2095-347X.2012.06.004]

8 Starov O, Dahse J, Ahmad S S, *et al*. No honor among thieves: A large-scale analysis of malicious web shells. Proceedings of the 25th International Conference on World Wide Web. Montréal, QB, Canada. 2016. 1021–1032.

9 胥小波, 聂小明. 基于多层感知器神经网络的 WebShell 检测方法. 通信技术, 2018, 51(4): 895–900. [doi: 10.3969/j.issn.1002-0802.2018.04.028]

10 毛勇, 周晓波, 夏铮, 等. 特征选择算法研究综述. 模式识别与人工智能, 2007, 20(2): 211–218. [doi: 10.3969/j.issn.1003-6059.2007.02.012]

11 周屹, 冯兆祥, 白熙卓, 等. 基于随机森林算法的数据分析软件设计. 黑龙江工程学院学报, 2017, 31(3): 38–41.