

智能检测 WebShell 的机器学习算法

戴桦, 李景, 卢新岱, 孙歆

(国家电网浙江省电力公司电力科学研究院, 浙江 杭州 310000)

摘要: WebShell 是网络入侵的常用工具, 具有危害性大、隐蔽性好等特点。目前的检测手段较简单, 容易被绕过, 难以对付复杂灵活的 WebShell。针对这些问题, 提出一种智能检测 WebShell 的机器学习算法, 通过对已知存在 WebShell 和不存在 WebShell 的页面进行特征学习, 完成对未知页面的预测, 灵活性、适应性较好。实验证明, 相比传统的 WebShell 检测方法, 该算法的检测效率、正确率更高, 同时也能以一定概率检测出新型的 WebShell。

关键词: WebShell 检测; 矩阵分解; 特征训练

中图分类号: TP309

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2017.00126

Machine learning algorithm for intelligent detection of WebShell

DAI Hua, LI Jing, LU Xin-dai, SUN Xin

(Electric Power Research Institute of Zhejiang Electric Power Corporation, Hangzhou 310000, China)

Abstract: WebShell is a common tool for network intrusions, which has the characteristics of great harm and good concealment. The current detection method is relatively simple, and easy to be bypassed, so it is difficult to deal with complex and flexible WebShell. To solve these problems, a supervised machine learning algorithm was put forward to detect WebShell intelligently. By learning the features of existing WebShell and non-existing WebShell pages, the algorithm can make prediction of the unknown pages, and the flexibility and adaptability were both very good. Compared with the traditional WebShell detection methods, the experiment proves that the algorithm has higher detection efficiency and accuracy, and at the same time there is a certain probability to detect new types of WebShell.

Key words: WebShell detection, matrix decomposition, feature training

1 引言

如今, 高速发展的互联网已经深入社会生活的各个方面, 从根本上改变了人们的生活方式, 网络环境为信息共享、信息服务创造了理想空间, 网络技术的迅速发展和广泛应用为人类社会的进步提供了巨大推动力。因此, 网络空间的安全性问题也逐渐成为网络建设的第一要素, 没有安全

保障的网络可以说是一座空中阁楼, 一旦脱离了计算机网络, 整个社会的运转就面临着崩溃。作为计算机网络中的核心, 服务器的安全性是整个计算机网络安全性和稳定性中至关重要的部分。

近年来, 12306、360、CSDN、京东、支付宝等各大网站相继出现用户信息泄露事件, 其中绝大部分都与 Web 网站安全息息相关, 尤以网站后门的安装最为频繁。网站后门又被称为 WebShell,

收稿日期: 2016-10-31; 修回日期: 2016-12-08。通信作者: 戴桦, yangmingyuan@yuantiaotech.com

基金项目: 国家自然科学基金资助项目 (No.61472113); 浙江省杰出青年基金资助项目 (No.LR14F020003)

Foundation Items: The National Natural Science Foundation of China (No.61472113), The Outstanding Youth Fund of Zhejiang Province (No.LR14F020003)

是一种基于 Web 服务的后门程序。在服务器安全领域中, WebShell 常作为获得服务器权限的一把利器, 为广大入侵者所使用。

WebShell 是一种以 asp、php 和 jsp 等网页文件形式存在的命令执行环境。入侵者可以通过 xss、sql 注入等手段获得上传权限, 再将包含 WebShell 的网页文件上传至服务器, 之后入侵者可以访问该网页文件, 通过 WebShell 获得更高的权限, 以达到控制服务器的目的。根据 WebShell 的大小和功能, 可以将 WebShell 分成大马、小马和一句话木马。大马功能全面, 一般包括提权命令、数据库连接、磁盘管理等功能, 入侵者可以进行文件和数据库的删除和修改, 或执行一些命令等; 小马体积小、隐蔽性强, 但所拥有的功能较少, 通常只能用于文件上传以及为数据库提权; 总之, 木马是指仅有一行代码完成的可动态接收脚本指令并执行的脚本木马, 多用于代码执行一些关键函数, 由于该木马通常被插入原有的正常文件中, 代码量极小, 因而隐蔽性强, 不易被检测发现。

WebShell 通常具有以下 2 个特点^[1]。

1) 隐蔽性。WebShell 一般隐藏在正常文件中, 并修改文件时间以达到隐蔽性, 或利用服务器漏洞进行隐藏, 使管理员很难发现这些 WebShell。

2) 容易穿越防火墙。由于与被控制的服务器远程通过 80 端口传递, 因此不会被防火墙和杀毒软件拦截。此外, WebShell 一般不会对系统日志中留下记录, 只会在网站的 Web 日志中留下一些数据提交记录, 因此, 管理员很难看出入侵痕迹。

传统的 WebShell 检测方法包括静态检测、动态检测、统计分析等多种方法, 每种方法分别针对特定类型的 WebShell, 具有一定的检测能力, 但都有各自的局限性, 无法形成统一有效的解决方法。为解决传统方法存在的问题, 本文提出一种基于矩阵分解的 WebShell 检测手段, 利用机器学习的方法, 让算法从训练数据中自动学习其中的特征, 进而使用学习之后的算法对数据进行预测, 最终达到分类的目的。与现有的一般 WebShell 检测方法和其他利用机器学习进行分类的方法相比, 本文所使用的方法放弃了对预测结果的可解

释性, 进而追求更准确的预测精度; 另外, 在通过训练过程后进行预测的速度非常快。

2 相关工作

为保证 WebShell 不被服务器的管理员发现, 可以长时间被入侵者利用, 入侵者需要在上传 WebShell 的同时对其采取如下的一系列保护措施^[2]。

1) 插入注释。在 WebShell 代码中插入一些注释, 这些注释不会影响 WebShell 的正常功能, 但会对检测工具形成干扰, 从而在一定程度上对 WebShell 进行保护。

2) 代码加密。对控制代码(或数据)在发送或使用前进行加密, 在运行前再进行解密。该方法因为将页面的特征混淆, 机器也无法对加密代码进行识别, 因而可有效地针对静态检测方法。

3) 多重编码。与代码加密相似, 这里对特征函数或参数进行多重编码, 使页面存在的一些规律发生改变, 降低被检测到的概率。

4) 外形混淆。入侵者在自己的 WebShell 中加入大量注释、无用变量或函数等, 减少核心代码占整个页面的比例, 使管理员误解该页面功能, 混淆管理员的检测。安全工具也会在检测过程中受其影响。

5) 字符串拆分。在不影响功能的前提下, 入侵者将函数名、变量名、常量名等标识进行分拆, 在使用的时候再将名字拼接起来以起到躲避检测的作用。

6) 页面分拆。为了避免单个文件中调用的关键函数或关键参数过于密集而被检测发现, 将一个完整页面拆分成多个页面上传, 在使用时采用 include 等函数调用其他页面中的关键函数或参数。该方法由于降低了关键函数或参数的比例, 可有效降低被检测到的概率。

针对以上问题, 管理员一般采用以下方法来预防或检测 WebShell。

1) 静态特征检测。静态特征检测方法是针对脚本文件的特征来进行的, 这些特征一般包括关键词、高危函数、文件权限、所有者等多维度信息。如果设定的特征比较合理, 那么该方法的检测效果好、成功率高, 可以很快检测出 WebShell。叶飞等^[3]针对页面的结构特征和文本特征进行分

类, 使用词袋模型提取关键词, 最后使用支持向量机的方法检测 WebShell; 胡建康等^[4]提取目标页面的不同特征, 然后使用决策树的方法对页面进行分类来检测 WebShell 的存在。

但该方法具有明显的缺点, 该方法只能对一些已有的 WebShell 有效, 对一些 0day 的 WebShell 基本无法检测。此外, 该方法容易产生误报, 无法对经过加密等特殊处理的 WebShell 文件进行检测, 也无法对窃密型的 WebShell^[5]做到准确检测, 因为窃密型 WebShell 的特征和正常的 Web 脚本十分相似。现在, 静态检测一般会与手动检测互相配合完成 WebShell 的防御, 然而对主机数量较多的公司或网站来说还存在耗费时间成本等问题。

2) 动态检测。动态检测方法是对在 BS 活动中产生的流量请求和响应、系统命令以及状态变化进行检测, 发现异常行为或状态的发生并最终检测 WebShell 的存在。例如, 检测到存在一个用户访问或调用一个从未使用过的文件, 那么该文件中存在 WebShell 的概率大大增加, 即从行为模式上深层次解析 WebShell 的存在。该方法对新型 WebShell 有一定的检测能力, 但对一些特定的后门难以发觉, 部署难度大; 入侵者也可以针对动态检测的存在将 WebShell 放入已有代码中, 加大动态检测的难度。

3) 统计分析^[6]。基于统计学的 WebShell 检测方法针对用户的访问特征, 通过统计学方法统计出这些特征的正常范围, 并将其与用户上传的脚本文件进行比较, 最终判断出 WebShell 的存在。该方法对经过编码和加密的 WebShell 依然有效, 因为这些 WebShell 也会表现出一些特别的统计特征。一般有重合指数、信息熵、最长单词长度、压缩比等方面的统计学分析技术。该方法一般用于识别混淆、加密的代码, 在识别模糊代码或混淆编排的木马方面表现良好。但也存在明显的缺点, 未经模糊处理的代码对统计检测方法较透明, 如果代码整合于其他脚本中, 则很有可能被认为是正常文件。

此外, 常用的检测方法还有: 基于流量分析的检测技术^[7], 检测 WebShell 在访问过程中产生的网络流量; 基于行为分析的检测技术, 通过分

析文件对系统所做的一系列操作来判断是否为 WebShell; 日志分析技术, 通过大量的日志文件建立请求模型检测异常文件; 词汇分析技术^[8]。

3 基于矩阵分解的 WebShell 检测算法

3.1 算法流程

本文用 u 对应矩阵某一列的特征, i 对应矩阵某一行的特征。 $r_{ui} \in [0, 1]$ 代表对象 u 对应对象 i 的评分, r_{ui} 越接近 1, 表示特征 u 对应特征 i 越可能存在 WebShell; 反之, r_{ui} 越接近 0, 表示对象 u 对应对象 i 越不可能存在 WebShell。 R 对应 $m_u \times m_i$ 的评分矩阵, 该矩阵拥有 m_u 列和 m_i 行。此外, 由于在大部分情况下, 评分矩阵 R 中的绝大部分元素为空, 所以本文使用三元组 $s = (u, i, r_{ui})$ 代表一条评分信息, 所有信息保存在数据集 $S = \{(u, i, r_{ui}) | r_{ui} \text{ 已知}\}$ 中。

本文使用矩阵分解模型将每个列向量 u 的 k 维因子向量 $p_u \in R^k$, 每个行向量 i 的 k 维因子向量 $q_i \in R^k$ 所结合^[9]。令 $\hat{r}_{ui}$ 表示对 r_{ui} 的预测, 通过式(1)得到预测值

$$\hat{r}_{ui} = p_u^T q_i \quad (1)$$

另外, 本文在模型中加入了向量的惩罚值, 减少参数值, 以避免模型出现过拟合的情况^[10]。给定一个评分矩阵的训练集 $S_i \subset S$, 通过式(2)对模型的参数进行估计。

$$\underset{\Theta}{\operatorname{argmin}} \sum_{s \in S_i} (r_{ui} - \hat{r}_{ui})^2 + \lambda (\|p_u\|_2 + \|q_i\|_2) \quad (2)$$

本文研究的基于矩阵分解的 WebShell 检测算法采用的是一种有监督的机器学习算法, 通过对已知页面数据和特征进行分析, 得出特征的权重, 并由此建立一个矩阵模型得出对未知页面是否存在 WebShell 的预测。该算法的流程如图 1 所示。

3.2 特征训练

从上文对矩阵分解的基本定义可以看出, 一般的矩阵分解模型是由不同的特征 u 对不同的特征 i 进行评分预测的, 所以这里所需要关注的变量特征只有 2 种, 即特征 u 和特征 i 。例如, 可以把 u 当成是用户, 这样不同的列向量代表不同的用户; 把 i 当成物品, 这样不同的行向量代表不同的物品。

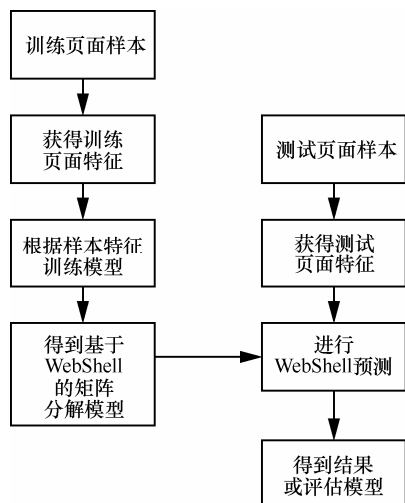


图1 基于矩阵分解的WebShell检测算法流程

但在WebShell的检测过程中,由于需要检测的变量特征数量较多,如单词数量、不同单词的数量、最大单词长度等,个数要远大于2,如果仍然使用传统的矩阵分解模型对WebShell进行预测,如用特征 u 代表单词数量,不同列向量代表不同单词的数量,由于特征太多,如果将矩阵分解成为多个矩阵相乘,这样难度太大;况且单词数量的上限并不确定,无法确定列的数量,因此使用传统的方法是不可行的,所以,需要对模型的特征进行一定修改。

在对模型特征进行修改之前,首先需要确定在WebShell进行检测过程中所需要关注的特征究竟有多少种。在特征选择^[11,12]的时候,如果选择的特征数量不足,则在评分预测过程中可能会有缺失,会对评分产生较大影响,进而影响评分预测的准确性;而如果选择的特征数量过多,则会使模型出现过拟合的情况,这样在对全新的内容进行预测时会产生偏差较大的情况,也会使模型变得更加复杂,所花费的时间也会变长。所以应该选择一个个数合适且在预测过程中权重较大的特征集合。本文将特征分为文本特征和其他特征,其中,文本特征为检测对象在文本内容方面所具有的特征,包括文本长度、包含单词种类、最长单词长度等;其他特征指除文本特征外的特征,包括文件操作、函数操作等。本文选取如表1所示的特征^[13]。

在确定特征数量后,需将这些特征转换成矩阵分解模型可用的形式。本文采用分类组合的方法。

表1 WebShell检测模型特征

特征类型	特征描述
文本特征	单词数量
	不同单词的数量
	最大单词长度
	文本总长度
	注释数量
其他特征	特殊字符数量
	字符操作函数调用
	关键函数调用
	加解密函数调用
	系统函数调用
	文件调用
	ActiveX 控件调用
	数据库调用
	脚本数量

首先,将文本特征中的每个特征进行分组,例如,特殊字符数量分成0~10个、10~50个、50个以上这3组;最大单词长度分为0~10个、10~20个、20个以上这3组;以此类推,具体分组的界限和分组的个数可以根据不同情况进行调整。在对文本特征中的6种特征进行分组后,从每种特征中随机选择一种进行组合。如果每种特征可以分成3组,就可以从文本特征中得出 3^6 ,共729种排列组合。

以此类推,本文对其他特征中的每种特征也进行分组。如果每种特征可以分成两组,则可以从其他特征中得出 2^8 ,共64种排列组合。

在得到这些排列组合后,本文将这729种不同的文本特征组合当作特征种类中不同的特征,将这64种不同的其他特征组合当作特征种类中不同的特征,这样就确定了评分矩阵 R 中列向量和行向量的数量以及每个列向量和行向量对应的含义。这样,评分矩阵 R 就是一个 729×64 的矩阵。

3.3 矩阵分解

在确定了评分矩阵及其对应的列向量、行向量后,就可以使用之前介绍的矩阵分解模型^[14]进行训练和预测。

在进行WebShell训练过程中,对于一个给定的文件, $\hat{r}_{ui}$ 作为因变量,代表该文件中是否存在WebShell。其中, u 代表该文件的文本特征组合情况,该文件的文本特征与哪一列对应的文本特征组合相同,该文件就与哪一列对应; i 代表该文件的其他特征组合,该文件的其他特征与哪一行对应的其他特征组合相同,该文件就与哪一行对应。这样每一个给定的文件都可以与每一个矩阵中的一个元素所对。因此,存在式(3)的方程。

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T q_i \quad (3)$$

在式(3)中, μ 代表评分矩阵的所有预测的平均值, 参数 b_u 和 b_i 代表文本特征组合 u 和其他特征组合 i 对应存在 WebShell 可能性的预测平均值。而文本特征组合 u 的 k 维因子向量 $p_u \in R^k$ 与其他特征组合 i 的 k 维因子向量 $q_i \in R^k$ 为参数向量, 代表了对应文本特征组合或对应其他特征组合的参数值, 该参数值随着训练过程会不断地进行变化和调整。本文使用梯度下降的方式对 p_u 和 q_i 进行训练。

为了使经过训练后的损失函数最小, 得到更准确的预测结果, 即

$$\operatorname{argmin}_{\Theta} \sum_{(u,i) \in S_t} (r_{ui} - \hat{r}_{ui})^2 + \lambda (b_u^2 + b_i^2 + \|p_u\|_2 + \|q_i\|_2) \quad (4)$$

训练过程如下。

- 1) 对训练集中每一个给定的结果 r_{ui} , 通过式(3)进行预测得到 $\hat{r}_{ui}$ 。
- 2) 由 $\epsilon_{ui} = r_{ui} - \hat{r}_{ui}$ 计算出预测误差。
- 3) 根据梯度下降公式对 p_u 和 q_i 进行更新。

$$p_u \leftarrow p_u + \gamma (\epsilon_{ui} q_i - \lambda p_u) \quad (5)$$

$$q_i \leftarrow q_i + \gamma (\epsilon_{ui} p_u - \lambda q_i) \quad (6)$$

其中, γ 是梯度下降算法的步长, λ 为学习速率。

在得到经过训练之后的模型后, 就可以利用该模型进行 WebShell 的预测, 并通过预测结果与真实结果的比较对该模型进行评判。对于一个需要进行预测是否存在 WebShell 的文件, 本文首先分析表 1 中提到文件的相关特征, 通过文件的特征确定文件对应的行和列及其对应的 p 和 q , 这里的 p 和 q 即经过训练的对应该参数。然后, 通过式(3)即可确定预测该文件对应的结果。若结果大于 0.5, 则该文件偏向于存在 WebShell, 且数值越大, 存在的可能性越大; 若结果小于 0.5, 则该文件偏向于不存在 WebShell, 且数值越小存在的可能性越小。

4 实验

4.1 实验数据

对于不同的语言, 如 php、asp、jsp 等, 根据这些语言所编写的 WebShell 脚本相似度较高, 并且本文基于矩阵分解的 WebShell 检测算法并没有针对语言上的不同而发生什么变化, 所以本文收集了 php 语言的 WebShell 作为实验数据。本文总共使用了 1 200 个不存在 WebShell 的 php

页面和 447 个存在 WebShell 的 php 页面。在进行 WebShell 检测算法的训练和测试前, 本文将页面分成 5 份, 进行 5 轮交叉试验, 选择 4 份页面作为训练数据, 另外 1 份作为测试数据, 计算 5 轮交叉试验的平均结果, 借此评判模型的检测能力。

在对页面特征进行分类时, 对于结果为两类的特征, 本文直接按照结果的不同将特征分成两类; 而对于结果为数字的特征, 本文按照特征结果的众数分类, 即假设该特征结果的众数为 p , 最大值为 n , 则将特征结果分成 $\left[0, \frac{p}{2}\right]$ 、

$\left(\frac{p}{2}, \frac{n-p}{2}\right]$ 、 $\left(\frac{n-p}{2}, \infty\right)$ 这 3 类, 对训练和预测

的数据按照这个分类标准进行分组。在参数选择时, 经过测试后, 本文选择步长 γ 为 0.1, 学习速率 λ 为 0.05。

4.2 评判标准

为了对本文中介绍的矩阵分解预测模型进行评测, 本文采取了 2 种方式进行评测。

1) 均方根误差, 又称为标准误差 (RMSE), 是观测值与真值偏差的平方与观测次数 $|S_t|$ 比值的平方根。

$$RMSE(S_t) = \sqrt{\frac{1}{|S_t|} \sum_{(u,i) \in S_t} (r_{ui} - \hat{r}_{ui})^2} \quad (7)$$

RMSE 越小, 说明预测的误差越低, 结果越好。

2) 准确率、召回率、 F 值评测^[15]。准确率是指正确预测为 WebShell 的数量占有所有预测数量中的比例, 召回率是指正确预测为 WebShell 的数量占有所有 WebShell 数量的比例。表 2 为预测结果分类, 其中, 0 代表 False(Negative), 1 代表 True(Negative), True Negative(TN)表示实际结果为 True, 而预测结果为 Negative, 则预测的准确率为 $\frac{TP + TN}{TP + FP + TN + FN}$, 召回率为

$\frac{TP}{TP + FN}$, 计算可得 F 值为 $\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$ 。准

确率、召回率以及 F 值越高, 说明预测的结果越好。

表 2 预测结果分类

实际结果	预测结果	
	0	1
0	False Negative (FN)	False Positive (FP)
1	True Negative (TN)	True Positive (TP)

4.3 实验结果

本节对本文提出的基于矩阵分解的 WebShell 检测方法的性能进行评估,说明本文采取的机器学习算法相比其他算法可以在 WebShell 检测中有更好的效果,所使用的比较算法为 Logistic 算法^[16]。同时本文也使用市面可见的某安全产品作为 WebShell 检测评估中的基础结果。

Logistic 算法是一种广义的线性回归算法。这里假设存在 m 个独立变量 $\mathbf{x}=(x_1, x_2, \dots, x_m)$, 又有条件概率 $P(y=1|x)=p$ 为某事件 P 发生的概率, 那么该算法可以表示为

$$P(y=1|x)=\frac{1}{1+e^{-g(x)}} \quad (8)$$

其中,

$$g(x)=\boldsymbol{\theta}^T \mathbf{x} \quad (9)$$

在式(9)中, $\mathbf{x}=(x_1, x_2, \dots, x_m)$ 对应表 1 中文档各个属性的数值, $\boldsymbol{\theta}$ 为各个属性对应的参数, Logistic 算法将文档属性与数值的内积映射到区间 $[0,1]$ 中, 以此来预测 WebShell 是否存在。与本文之前介绍的基于矩阵分解的 WebShell 检测算法相同, 概率 $p>0.5$, 则预测该文件中存在 WebShell; 概率 $p<0.5$, 则预测不存在 WebShell。

实验结果如表 3 和表 4 所示。

表 3 WebShell 的 RMSE 预测结果

检测方法	RMSE
矩阵分解	0.211
Logistic	0.238
某安全产品	0.520

表 4 WebShell 的准确率、召回率以及 F 值预测结果

检测方法	准确率	召回率	F 值
矩阵分解	0.982	0.813	0.890
Logistic	0.880	0.601	0.714
某安全产品	0.497	0.411	0.450

从实验结果中可以看出, 矩阵分解和 Logistic 对 WebShell 检测的 RMSE 结果和准确率结果都比某安全产品好, 说明利用机器学习的方法进行 WebShell 的检测是有效的, 可以提高 WebShell 检测的成功率。

此外, 本文也把矩阵分解的方法和 Logistic 算法进行比较, 发现利用矩阵分解进行 WebShell

检测的方法要略好于 Logistic 算法, 出现这种情况, 推测有 2 种原因: 首先, 本文使用的矩阵分解算法更为先进, 相比于 Logistic 算法, 可以更好地从页面信息中挖掘出 WebShell 的特征并进行更为准确的预测; 其次, 在数据使用方面, 本文提出的使用矩阵分解进行 WebShell 检测的方法是将页面的特征数据分类后使用的, 而 Logistic 算法则是使用准确的特征数据, 这可能是造成 WebShell 检测差异的原因之一, 因为 Logistic 使用极为精确的特征数据可能会造成过拟合的出现, 而对特征数据进行分类后使用则会将一定范围内的特征数据看成是相同的数据, 在一定程度上会减缓过拟合的情况, 从而使 WebShell 检测结果更加准确。

5 结束语

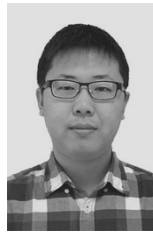
本文分析总结了 WebShell 的特点、分类、隐藏方法及相应对策, 每种检测方法都有一定的局限性。为解决现有 WebShell 检测方法的问题, 本文提出了一种基于机器学习的 WebShell 检测方法, 通过让算法自学习包含 WebShell 和不包含 WebShell 的页面, 来达到智能检测的目的, 克服了传统基于特征匹配、流量检测等方法的不足, 提升了 WebShell 检测的准确率和召回率。此外, 本文算法的输入数据源越多, 学习效果越好, 检测效果也会越好, 实验中使用了 1 647 份页面, 正确率可达 98.2%, 可以看出本文算法的显著效果。然而, 在对页面特征进行分类时, 分类方法的合理性和有效性并没有得到证实, 希望在之后的研究中可以找到一种准确的特征分类方法, 进一步提高 WebShell 检测的成功率。

参考文献:

- [1] MENG Z, MEI R, ZHANG T, et al. Research of Linux WebShell detection based on SVM classifier[J]. Netinfo Security, 2014.
- [2] MINGKUN X, XI C, YAN H. Design of software to search ASP WebShell[J]. Procedia Engineering, 2012, 29: 123-127.
- [3] 叶飞, 龚俭, 杨望. 基于支持向量机的 WebShell 黑盒检测[J]. 南京航空航天大学学报, 2015, 47(6): 924-930.
YE F, GONG J, YANG W. Black box detection of WebShell based on support vector machine[J]. Journal of Nanjing University of Aeronautics and Astronautics, 2015, 47(6): 924-930.
- [4] 胡建康, 徐震, 马多贺, 等. 基于决策树的 WebShell 检测方法研究[J]. 网络新媒体技术, 2012(6): 15-19.

- HU J K, XU Z, MA D H, et al. A decision tree based research on Webshell detection method[J]. Network and New Media Technology, 2012(6): 15-19.
- [5] 马立军. 基于行为监测的窃密型木马检测研究[J]. 广西民族大学学报, 2014, 20(2): 70-74.
- MA L J. A behavior monitoring based theft Trojan detection[J]. Journal of Guangxi University for Nationalities, 2014, 20(2): 70-74.
- [6] 石刘洋, 方勇. 基于 Web 日志的 Webshell 检测方法研究[J]. 信息安全研究, 2016(1).
- SHI L Y, FANG Y. Webshell detection method research based on Web log[J]. Information Security Research, 2016(1).
- [7] 马艳发. 基于 WAF 入侵检测和变异 WebShell 检测算法的 Web 安全研究[D]. 天津: 天津理工大学, 2016.
- MA Y F. Web security research based on WAF intrusion detection and mutation WebShell detection algorithm[D]. Tianjin: Tianjin University of Technology, 2016.
- [8] DENG L Y, DONG L L, CHEN Y H, et al. Lexical analysis for the WebShell attacks[C]//The International Symposium on Computer, Consumer and Control, IEEE Computer Society. 2016:579-582.
- [9] MURPHY K P. Machine learning: a probabilistic perspective[M]. Massachusetts: MIT press, 2012.
- [10] BISHOP C M. Pattern recognition[J]. Machine Learning, 2006, 128.
- [11] HOU Y T, CHANG Y, CHEN T, et al. Malicious Web content detection by machine learning[J]. Expert Systems with Applications, 2010, 37(1):55-60.
- [12] 路永和, 李焰锋. 改进 TE-IDF 算法的文本特征项权值计算方法[J]. 图书情报工作, 2013, 57(3):90-95.
- LU Y H, LI Y F. An improved TE-IDF algorithm for calculating the weight of text feature[J]. Library and Information Service, 2013, 57(3):90-95.
- [13] KHALID S, KHALIL T, NASREEN S. A survey of feature selection and feature extraction techniques in machine learning[C]//Science and Information. 2014:372-378.
- [14] 彭行雄, 肖如良, 张桂刚. 基于自适应提升的概率矩阵分解算法[J]. 计算机应用, 2015, 35(12):3497-3501.
- PENG X X, XIAO R L, ZHANG G G. Probabilistic matrix factorization algorithm based on Ada Boost[J]. Computer Application, 2015, 35(12):3497-3501.
- [15] 李航. 统计学习方法[M]. 北京:清华大学出版社, 2012.
- LI H. Statistical learning method[M]. Beijing: Tsinghua University Press, 2012.
- [16] SCHEIN A I, UNGAR L H. Active learning for logistic regression: an evaluation[J]. Machine Learning, 2004, 68(3):235-265.

作者简介:



戴桦(1985-), 男, 浙江杭州人, 硕士, 国家电网浙江省电力公司电力科学研究院工程师, 主要研究方向为网络安全、工业控制系统信息安全。



李景(1983-), 男, 山东泰安人, 硕士, 国家电网浙江省电力公司电力科学研究院高级工程师, 主要研究方向为电力信息化及网络信息安全。



卢新岱(1988-), 男, 浙江奉化人, 硕士, 国家电网浙江省电力公司电力科学研究院工程师, 主要研究方向为信息安全及工控安全。



孙歆(1981-), 男, 浙江杭州人, 硕士, 国家电网浙江省电力公司电力科学研究院高级工程师, 主要研究方向为应用系统安全、移动安全、工控安全。