基于多类特征的 JavaScript 恶意脚本检测算法*

付垒朋 张 瀚 霍路阳

(南开大学 计算机与控制工程学院 天津 300071)
(南开大学 天津市智能机器人技术重点实验室 天津 300071)

摘 要 针对脚本样本集具有混淆、统计、语义等不同层面上的特征,设计基于多类特征的 JavaScript 恶意脚本检测算法,实现针对恶意 JavaScript 脚本的离线分析系统 JCAD. 首先提取脚本的混淆特征,使用 C4.5 决策树分析被混淆的脚本并解除混淆. 然后提取脚本的静态统计特征,根据语义进行脚本序列化,构造危险序列树,提取脚本的危险序列特征. 最后以三类特征作为输入,采用对脚本样本集的非均匀性与不断增加的特点具有较强适应能力的概率神经网络构造分类器,判断恶意脚本. 实验表明,该算法具有较好的检测准确率与稳定性.

关键词 恶意脚本检测, 多类特征, 概率神经网络, 网页挂马攻击
中图法分类号 TP 389.1 **DOI** 10.16451/j.cnki.issn1003-6059.201512007

JavaScript Malicious Script Detection Algorithm Based on Multi-class Features

FU Lei-Peng, ZHANG Han, HUO Lu-Yang
(*College of Computer and Control Engineering*, *Nankai University*, *Tianjin* 300071)
(*Tianjin Key Laboratory of Intelligent Robotics*, *Nankai University*, *Tianjin* 300071)

ABSTRACT

Aiming at features of different levels in the script sample set, such as obfuscation, statistics and semantics, a malicious JavaScript script detection algorithm based on multi-class feature is proposed. The JavaScript analysis system, JavaScript codes analysis and detection, is implemented. The obfuscation features of the JavaScript are extracted and the obfuscated scripts are analyzed and deobfuscated by C4.5 algorithm. The static statistical features of the JavaScript are extracted, and according to the semantics, the JavaScript is serialized. Dangerous sequence tree is generated by the proposed algorithm to extract the dangerous sequence features of the malicious JavaScript. Three types of features are used as the input. The probabilistic neural network with strong ability to adapt to non-uniformity and the increasing quantity of the input samples is applied to construct the classifier for the detection of malicious JavaScript. The experimental results show that the proposed algorithm has better detection accuracy and stability.

* 国家自然科学基金项目(No. 61004086)、天津市自然科学基金项目(No. 15JCYBJC18900)资助
收稿日期:2014-10-22;修回日期:2015-01-20
作者简介 **付垒朋**,男,1983 年生,硕士研究生,主要研究方向为信息安全. E-mail:eldhj@163.com. **张瀚**(通讯作者),男,1978 年生,博士,教授,主要研究方向为计算智能、信息安全. E-mail:zhanghan@nankai.edu.cn. **霍路阳**,男,1989 年生,硕士研究生,主要研究方向为信息安全.

Key Words　Malicious Script Detection, Multi-class Features, Probabilistic Neural Network, Drive-by-Download Attack

1　引　言

JavaScript[1]是实现用户交互、浏览器控制和动态 HTML 内容创建的脚本语言,恶意 JavaScript 代码利用安全漏洞,可能获得浏览器的控制权,其中网页挂马(Drive-by-Download) 攻击是传播病毒,例如木马、后门等的常用机制. 同时,混淆技术[2]的使用可逃脱基于正则表达式、规则匹配及文件签名等反病毒技术查杀.

近年来,JavaScript 恶意脚本代码的检测成为焦点问题. Likarish 等[3]设计基于脚本统计的混淆特征的 JavaScript 恶意脚本分类器,因无法解混淆,检测率、误判率不理想. Egele 等[4]检测堆溢出攻击中常用的 shellcode 代码. Fraiwan 等[5]通过频繁词、函数、执行等代码特征构建分类模型. Blanc 等[6-7]将混淆化的脚本代码转化为中间语言处理,缺乏中间语言的进一步分析. 文献[8]和文献[9]建立抽象语法树,从代码语义分析角度进行恶意代码检测,由于只使用语义特征,分析较为单一. 上述静态分析方法分析网页脚本获取特征集,包括代码统计特征、危险对象、混淆化等,脚本并不执行,故速度较快,开销较小.

基于动态执行的分析方法需要模拟脚本运行以获取脚本代码执行特征. 低交互蜜罐工具有 JSAND[10]、PhoneyC[11]、JSUnpack[12]等. Curtsinger 等[13]将 JavaScript 检测器整合到浏览器的 JavaScript 解析器中,用户访问页面时实时检测,也使用抽象语法树提取特征实现检测系统 Zozzle. JSAND 系统利用 HtmlUnit 框架和脚本引擎 Rhino[14]模拟浏览器环境,监控脚本与 DOM 对象树的动态交互过程. Prophiler[15]是建立描述 Drive-by-Download 攻击中的 10 类特征的脚本代码执行规则库,无法检测新攻击规则的恶意代码,系统开销较大. 而高交互蜜罐 Capture-HPC[16]等是在基于虚拟机的真实操作系统中运行浏览器,虚拟机受攻击后恢复时资源消耗较大,漏检很多重要恶意脚本. Dewald 等将动态和静态分析方法结合,实现嵌入沙盒[17]的 Cujo 系统[18],使用机器学习对代码和代码的执行提取特征集,动态部分占用资源较大,速度较慢.

动态执行的脚本检测方法通常可实现较高的检测率,但系统资源占用量很大. 静态分析的检测方法速度较快,资源占用量较小,检测准确率依赖特征选择及分类器设计.

为避免现有动态脚本分析工具资源占用较大,不能实现大规模网页分析的缺点,本文使用静态分析,设计基于多类特征并使用概率神经网络(Probabilistic Neural Network, PNN)分类的恶意脚本检测方法,实现针对恶意 JavaScript 脚本的离线分析系统 JCAD (JavaScript Code Analysis and Detection),快速区分正常脚本代码和恶意脚本代码. 考虑普遍使用的 Drive-by-Download 攻击的特点,提取多种来源全面有效的特征集,使用混淆化特征判断脚本是否混淆,对混淆脚本进行解混淆,提取脚本的静态统计特征,由危险序列生成树提取危险序列特征,由脚本样本集的非均匀性、样本集不断增加的特点选择 PNN 训练分类器. 实验表明,该算法实现对 JavaScript 更准确的分析预测.

2　脚本特征的提取

特征的重要程度直接影响分类器的准确性,通过对 JavaScript 语言及 Drive-by-Download 攻击的充分分析,提取脚本的混淆化特征、静态统计特征和脚本危险序列特征等 3 类主要特征.

2.1　混淆特征

混淆技术是 Javascript 为隐藏编程函数细节设计的,混淆的脚本很难从字面理解且不易逆向分析,约有 50% 的杀毒软件将混淆化的代码认为是有害代码. 然而,正常网站如谷歌等为保护代码产权,也常使用混淆技术. 与之不同的是,有害脚本的混淆是为了逃脱病毒检测,而使用大量特殊的混淆技术.

文中将恶意脚本常用的混淆技术中提取的特征作为脚本特征集的一部分,同时对混淆化的脚本进行初步解混淆,以便提取其他特征.

考虑特殊格式进行重编码(base64 编码等)、全文加密、动态代码生成和执行操作 eval( ) 方法等特殊混淆技术,提取如下 9 类混淆特征:解密及编码用到的方法总数,包括 eval( ), escape( ), unescape( ), fromCharCode( ) 等,JavaScript 经常用这些方法实现字符串编码的转换;长度超过 128 bit 的字符串的个数,常由缓冲区溢出攻击引起;长度超过 256 bit 的行数,混淆化函数会去掉脚本代码的换行符;平均单词长度;平均行长度;空格所占的比例;可疑变形字符数,形如\u9090,%u4dfb,\x4a\x6d\x3f\x35 等;所

有字符串中数字所占的比例;脚本可读性比例[3].

2.2 静态统计特征

解混淆后的有害脚本与正常脚本在静态统计方面有着显著差异,如 shellcode 代码的出现、HTTP 请求特征和页面内容特征、危险对象等成为对未知有害代码检测的重要考察内容[4]. 脚本静态分析速度较快,选取区分恶意脚本的重要统计特征集尤为关键.

由于结构化语言 JavaScript 的关键字分布与自然语言有较大差异,自然语言的 unigrams 或 bigrams 分析方法会产生大量只存在恶意或正常脚本中且次数极少的无用特征,数量庞大且有效特征较少,这并不适用于结构化语言的特征提取[3].

考虑正常脚本与恶意脚本在结构方面的差异,通过区分正常脚本和恶意脚本的已有知识,提取如下 8 个统计特征:所有字符串中字母所占的比重;字符串总长度占脚本代码总数的比重;可疑编码总长度;可疑对象总数;长度超过 128 bit 的超长字符串占脚本长度的比重;字符串可疑操作总数,包括字符串连接、分割、编码、替换、转化等操作;文档对象操作总数,包括文档的 cookie 操作、URL 操作、文档流打开操作、向文档中写入 HTML 或 JavaScript 代码操作等;动态执行脚本操作总数、如 evel( )、document. createElement ( )、window. setTimeout ( )、window. setInterval( )等方法.

2.3 序列特征

Javascript 脚本代码由关键字、对象、函数等字符串单元组成,按各单元所属的语义类别编码可将脚本源代码按顺序分解为关键序列. 脚本静态语义可用此序列描述,文中将恶意脚本特有的序列特征称为危险序列特征,此特征模式可作为检测的重要内容. 字符串单元分类一般包括对象、关键方法、关键属性及其他自定义分类等,Drive-by-Download 攻击的常用序列包括可疑对象、字符串的分类、操作、异常字符串编码等,类别序列特征在很大程度上描述恶意代码中的特有语义.

本文重要的序列特征提取如下.

2.3.1 脚本代码序列化

定义字符串对象方法及属性、恶意脚本常用的对象、其他重要的方法及属性、字符串分类这 4 大类 32 种类别,具体分类如下.

1)字符串对象的方法及属性. 脚本混淆和恶意脚本经常使用字符串变换技术,涉及到字符串的查询、连接、分割、重编码、替换及转化等 6 种. 例如,字符串的查询操作,包括 charAt( )、charCodeAt( )、length( )、evec( )等,分别是查询特定位置的字符,查询字符串的长度及查询字符串符合正则表达式的子串,序列号记为 STR_INQUIRE_ID.

2)恶意脚本常用的对象. 主要有 ActiveX 对象、Error 对象、FileSystemObject 对象、正则表达式对象的生成操作、新函数对象、脚本对象的声明操作 6 种. 如脚本对象的声明操作,使用通用对象声明方法,序列号记为 NEW_OBJECT_ID.

3)其他重要的方法及属性类. 主要有 Window 对象执行代码、Location 对象网页链接操作、Location 对象加载文档操作、FSO 的查询及读取等操作、FSO 的修改操作、Document 对象的 Cookie 操作、Document 对象的 URL 操作、Document 对象的打开操作、Document 对象的写操作、脚本代码执行操作 10 种. 如脚本代码执行操作类,包括 evel( )方法、Document 对象的 CreateElment( )方法等,序列号记为 CODE_EXE_ID.

4)字符串分类. 字符串按长度及编码不同分为 10 种,如长度大于等于 128 bit 的变型字符串,此类字符串长度异常,可疑程度较大,序列号记为 STRING_ENCODE_LARGE.

根据本文定义的 32 种类别,将脚本代码的字符串单元用所属类别的序列号表示,可将脚本代码序列化. 属于多个类别时,按照类别顺序依次列出.

2.3.2 危险序列树生成步骤

使用恶意脚本集和正常脚本集生成危险序列树,不依赖于特殊攻击样本. 先构造样本序列树,根据样本序列树合并算法等得到样本集序列树,再产生危险序列树.

1)样本序列树. 样本序列树是 n 阶树,根节点上无信息仅表示起点,叶节点记录其后出现事件的频率分布,生成后续事件向量. n 长度窗口在样本序列上移动,每个位置提取一个 n 长度子串,若子串已在树中,不处理;若子串不在树中,子串作为新分支加入序列树中. 根据脚本语义关联特点,提取 4 阶样本序列树,一个样本生成一棵样本序列树.

2)样本序列树合并算法. 将 A、B 两棵样本序列树合并成序列树 C 的合并算法如下,假设序列总数为 m.

(1)以 A 树作为初始 C 树,根节点为第一阶.

(2)从 C 树第二阶开始,按事件 1 到 m 遍历,当事件 F 只在树 A 中出现时,以 F 为根的子树加入到 C 树中;当事件 F 只在 C 树中出现,不操作;当事件 F 同时在 B 树和 C 树中出现,事件 F 对应的节点频率为 B、C 两树频率相加,然后对事件 F 节点的子节点

递归执行相同流程.

(3) 当事件 F 对应的节点为叶节点时,将两个叶节点的后续事件分布向量相加.

包含 k 个样本的恶意脚本集的所有序列树可使用上述算法合并,得到恶意样本集序列树. 每个非根节点频率数除以 k,得到该节点在恶意脚本集中出现的概率;叶节点的后续事件分布向量除以 k,得到后续事件概率向量. 同理可生成正常集序列树.

3) 危险序列树及其生成算法. 恶意集序列树反映特征序列在恶意脚本集中出现的概率及分布,正常集序列树反映特征序列在正常脚本集中出现的概率及分布. 文中选取恶意脚本集中出现较多,并且在恶意集序列树中出现频率远大于正常集序列树出现频率的序列作为危险序列.

设恶意集序列树 A,正常集序列树 B,危险序列树 C. 危险序列树生成算法如下.

(1)A 树作为初始 C 树,剪除概率小于 1‰ 的节点.

(2) 遍历 C 树中除根节点以外的所有节点,若节点 E 出现概率大于 3 倍正常序列集对应节点概率,标记为危险.

(3) 对于叶节点,如果后续事件概率向量中有大于1‰的项,且该概率大于3倍正常集序列树对应位置概率时,将其生成叶节点的子节点,并标记为危险.

(4) 非危险节点剪枝. 如果节点为非危险节点,且其子树中无危险节点,将以该节点为根节点的子树剪除.

最终生成的危险序列树,从根节点到任意一个标记为危险的节点的序列都是危险序列,共有 50 个,其中,长度为1 的危险序列8 个,长度为2 的危险序列19 个,长度为3 的危险序列14 个,长度为4 的危险序列 9 个.

2.3.3　危险序列特征提取

利用长度为 n 的滑动窗口将样本序列划分为一系列字串[19],映射到危险序列树中,统计样本中危险序列出现的频率,形成样本的危险序列树. 使用深度优先算法取得危险节点的频率值,生成一个长度为 50 的危险序列出现频率向量作为特征向量.

3　基于概率神经网络分类的检测系统

在静态分析技术基础上,设计基于多类特征与 PNN 的恶意脚本检测方法,并实现 JCAD,用此检测网页是否包含恶意 JavaScript 脚本代码.

3.1　概率神经网络原理与特点

由于恶意脚本集分布的不均匀性,表现为多个不同规模聚类的集合,使得 SVM 等机器学习算法对恶意脚本与正常脚本的区分度不高. 而 PNN[20] 可把任何输入映射到多个类别,新增数据可实时修改非线性判定边界,使之接近贝叶斯最佳判定面,实现网络的动态更新,适合处理不断增加的恶意脚本样本集. PNN 包含输入层、径向基神经元层和竞争神经元层. 给定训练样本,由学习算法确定隐含节点数、确定径向基函数中心及散布常数 *spread*、确定竞争神经元权值系数矩阵等参数,其中径向基神经元学习算法[21] 利用 k-means 对输入样本聚类,竞争神经元学习过程[22] 是网络权值调整的过程. 其优点是易训练,调整 *spread* 可增加判定面的形状复杂度,容许错误样本和信息噪声存在,并且不需要从各个神经元到输入神经元的反馈,可训练样本确定网络拓扑结构,自组织能力和自学习能力较强,对于非均匀性的恶意脚本样本集适应能力较强,在模式分类领域准确性较高,训练时间较短,权重变化的鲁棒性较好.

3.2　使用概率神经网络分类器的检测方法与系统架构

检测方法是由输入的 JavaScript 脚本代码获取脚本的混淆特征、静态统计特征和危险序列特征,得到特征向量,在训练 JCAD 分类器后,对待测脚本进行分类. JCAD 分类器分为训练学习阶段和测试阶段.

图 1 中,虚线方框内部表示系统主要部分 JCAD 分类器,训练算法步骤如下.

1) 由爬虫工具等方法获得训练样本.

2) 由所有训练样本得到混淆化组成的特征矩阵,生成 C4.5 决策树[23] 作为混淆化判别器.

3) 训练样本特征矩阵的提取. 首先,提取样本的混淆化特征,通过混淆化判别器判断脚本是否混淆化,如果含有混淆化代码,利用 JS Beautifier 工具[24] 解混淆化;然后,获取各个脚本文件的统计特征向量;同时启动危险序列特征获取模块.

4) 特征处理. 包括特征合并,归一化处理,降维,形成所有训练样本最终的特征矩阵.

5)PNN 分类器 JCAD 的训练与生成.

JCAD 分类器处理待测样本时,得到待处理样本的特征向量作为输入,得出是否恶意的分类结果.

混淆判别、危险序列特征提取、特征降维等模块处理后得到特征矩阵,用于训练与生成 PNN 分类器.

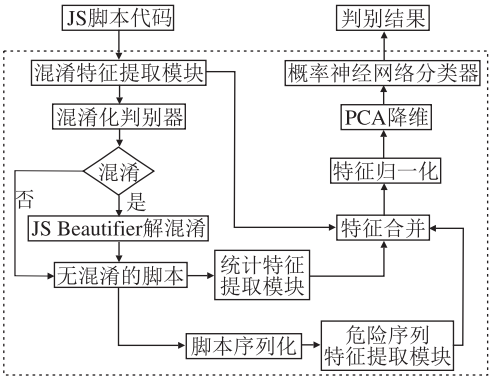

图 1 JCAD 分类器框架图

Fig. 1 Framework of JCAD classifier

3.3 混淆判别模块

图 1 中的混淆判别器由 C4.5 决策树[23] 生成,用于判断脚本代码是否混淆,步骤如下.

step 1 获取所有训练样本的混淆特征,形成样本混淆特征集 S,产生根节点,S 归到该节点下.

step 2 计算特征集 S 的信息熵 $Info(S)$:

$$Info(S) = -\sum_{i=1}^{2} \frac{c_i}{M} \log_2 \frac{c_i}{M},$$

其中,共有两类样本,表示为混淆文件与未混淆文件;c_i 为第 i 类样本数量;M 为样本总数.

step 3 每个样本的属性向量包含 n 个连续型的属性 $V_i, i = 1,2,\cdots,n$. 对于 V_i,使用 m 个划分点 $\gamma_1,\gamma_2,\cdots,\gamma_m$ 等量划分属性 V_i. 假设属性 V_i 的一个分裂点 $\gamma_i, i = 1,2,\cdots,m$,将特征集划分为 2 部分 S_1 和 S_2,数量为 M_1 和 M_2.

对数据集 $S_j, j = 1,2$. 属性 V_i 分别在 m 个划分点下的信息熵 $Info_{V_i}(S_i)$:

$$Info_{V_i}(S_i) = \sum_{i=1}^{2} \frac{M_i}{M} Info(S_i),$$

属性 V_i 分别在 m 个划分点下的信息增益 $Gain(V_i)$:

$$Gain(V_i) = Info(S) - Info_{V_i}(S),$$

属性 V_i 分别在 m 个划分点下的分裂信息值 $SplitInfo(S)$:

$$SplitInfo(S) = -\sum_{i=1}^{2} \frac{M_i}{M} \log_2 \frac{M_i}{M},$$

属性 V_i 分别在 m 个划分点下的信息增益率 $GainRatio_{V_i}(S)$:

$$GainRatio_{V_i}(S) = \frac{Gain(V_i)}{SplitInfo_{V_i}(S)}.$$

step 4 取属性 V_i 分别在 m 个划分点下的最大信息增益率作为属性 V_i 的信息增益率,并记下该划分 γ_j.

step 5 利用 step 3 和 step 4 计算所有属性的信息增益率,选择具有最大信息增益率的属性,将当前节点标记为该属性,记下属性划分值. 使用此节点,将样本特征集划分为 2 个子集 S_1 和 S_2.

step 6 分别判断集合 S_1 和 S_2 是否满足停止生长的条件,若满足,执行 step 7;否则,从 step 2 开始执行,递归生成一棵子树.

step 7 当某一类别样本所占比重达到预先设置的精度,或样本数目足够少时停止决策树生长,将该节点标记为叶节点,类别设置为优势类别. 最后,决策树剪枝.

将脚本的混淆化特征向量输入到混淆判别器中,生成从决策树的根节点到叶节点的一条判定路径,最终叶节点的类别即为待测样本的类别.

3.4 危险序列特征提取模块

按 2.3 节设计的算法提取危险序列特征,步骤如图 2 所示.

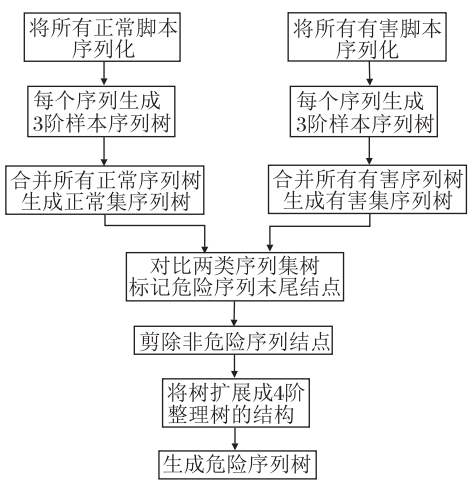

图 2 危险序列树生成步骤

Fig. 2 Steps for generating dangerous sequence tree

3.5 特征降维

本文提取的混淆特征数为 9 个,危险序列特征为 50 个,统计特征为 8 个,共 67 个特征. 为减少冗余,采用主成分分析法(PCA) 对样本特征进行线性降维[25],提取前 43 个主分量,积累主分量贡献率为 94.8%.

3.6　概率神经网络分类器模块

最关键的训练阶段是将训练样本特征矩阵作为输入,训练 PNN,包括参数选定、性能调优等. 设训练样本总数为 N,样本特征向量为 t 维. 分类器的训练与测试步骤如下.

step 1　已得到所有训练样本的特征矩阵 $\boldsymbol{Q}$ 和样本类别向量 $\boldsymbol{T}$,$\boldsymbol{Q}$ 为 $t \times N$ 矩阵,$\boldsymbol{T}$ 为 $1 \times N$ 向量. 把样本类别向量 $\boldsymbol{T}$ 转化成 PNN 识别的向量 $\boldsymbol{V}$.

step 2　选择最优的径向基函数的散布常数,它是PNN中的重要参数,选取步骤见4.2.1节,默认取0.1.

step 3　使用第 3.1 节所述学习算法与权值调整算法[21-22] 训练并构建 PNN 分类器.

step 4　在样本检测阶段,待测样本特征向量 $\boldsymbol{Z}$ 为 $1 \times N$ 维,用训练好的 PNN 分类器判别样本类别,输出 PNN 使用的目标类别.

step 5　分类结果以向量形式表示,样本归属类别对应的项为 1,其余项为 0,将目标类别转化为可识别类别,完成样本检测.

4　实验及结果分析

4.1　实验数据

使用版本 1.15.4 的 Heritrix 工具爬取网页获得部分资源,以反映脚本在真实网络的分布. 获取正常样本集包含 3 440 个正常脚本. 利用 Alexa 网站(www.alexa.com/topsites)的前 100 位网站 URL 作为种子,爬取网页文件. 经 JSAND[10] 分析与人工分析提取正常 JavaScript 脚本.

获取的恶意样本集包含 500 个恶意脚本,来源如下.

1)从安全网站如看雪论坛、剑盟、卡饭论坛、http://malwareurl.com、http://malwaredomainlist.com、http://tools88.com/safe/JSencode.php、http://www.malekal.com 等得到可疑 URL 列表作为种子,爬取 3 721个 JavaScript 代码,人工分析得到恶意脚本 201 个.

2)从 VX Heavens 网站下载源于卡巴斯基公司的 JavaScript 恶意脚本 263 个.

3)使用 Drive-by-Download 代码生成器,得到 36 个样本. 人工分析得到:正常脚本混淆率为 17.7%,恶意脚本混淆率为 83.6%,总混淆率为 26.1%,可见混淆在恶意代码中有重要地位.

4.2　实验结果分析

样本集中病毒集与正常集数量比例为 1∶6.88,为非平衡数据集. 采用如下 7 个常用指标评价分类器效果.

1)TP(True Positives):正常脚本判断为正常脚本的数量;

2)TN(True Negatives):恶意脚本判断为恶意脚本的数量;

3)FP(False Positives):恶意脚本判断为正常脚本的数量,即漏报数;

4)FN(False Negatives):正常脚本判断为恶意脚本的数量,即虚警数;

5)正常脚本判别正确率:

$$TPR = \frac{TP}{TP + FN};$$

6)恶意脚本判别正确率:

$$TNR = \frac{TN}{TN + FP};$$

7)可评价非平衡数据集中总体分类性能的几何均值:

$$G_{\text{mean}} = \sqrt{\frac{TN}{TN + FP} \cdot \frac{TP}{TP + FN}}.$$

后 3 个指标尤为重要. 为验证分类器的稳定性,采用交叉验证方式[26] 评估分类器.

4.2.1　散布常数对分类器性能的影响

径向基函数的散布常数 *spread* 是主要参数,即径向基函数的平滑因子,需选择合适取值使分类器准确度达到最优. 当 *spread* 接近 0 时分类器成为一个最近邻居分类器;*spread* 增大会成为最近 n 个邻居分类器,n 与散布常数正相关. *spread* 越小,径向基函数逼近越精确,但过程越不平滑,出现过适应现象,影响网络性能;*spread* 越大,径向基函数越平滑,但逼近误差越大,隐藏神经元越多,计算量越大. 理想的 *spread* 应尽量大,使神经元产生响应的输入范围可覆盖足够大的区域但不能过大,应使各个神经元都具有重叠的输入向量响应区域. 在实验中,使用 10 折交叉验证方法,将不同 *spread* 下的 10 次实验结果相加,评判在该 *spread* 条件下的分类准确度,结果如表 1 所示.

由表 1 可见,*spread* 从 0.01 到 0.50 时,分类器对正常脚本的检测率呈增长趋势,检测率都在 99.5% 以上. 对恶意脚本的检测率呈先增后减的趋势,在 *spread* 为 0.06 时达到最大的 91.8%,此时分类器的总体分类性能 G_{mean} 也达到最好,所以文中选

择 *spread* 为 0.06.

综上所述,分类器对正常脚本的检测率可达到 99.6%,对恶意脚本的检测率可达到 91.8%,总体分类性能 G_{mean} 可达 95.6%.

表 1 不同 *spread* 下的分类指标值

Table 1 Values of evaluation indicator for classifier performance with different values of *spread*

spread	*TP*	*TPR*	*TN*	*TNR*	G_{mean}
0.01	3425	0.9956	436	0.872	0.9317
0.02	3425	0.9956	454	0.908	0.9508
0.03	3425	0.9956	457	0.914	0.9539
0.04	3427	0.9962	457	0.914	0.9542
0.05	3427	0.9962	458	0.916	0.9553
0.06	3426	0.9959	459	0.918	0.9562
0.07	3426	0.9959	458	0.916	0.9551
0.08	3422	0.9948	457	0.914	0.9551
0.09	3422	0.9948	456	0.912	0.9525
0.10	3424	0.9953	452	0.904	0.9486
0.11	3424	0.9953	448	0.896	0.9443
0.12	3424	0.9953	439	0.878	0.9348
0.13	3424	0.9953	430	0.860	0.9252
0.14	3425	0.9956	422	0.844	0.9167
0.15	3426	0.9959	414	0.828	0.9081
0.20	3427	0.9962	371	0.742	0.8598
0.30	3433	0.9980	312	0.624	0.7891
0.50	3440	1.0000	160	0.320	0.5657

4.2.2 概率神经网络算法与其他机器学习算法对比

选择机器学习算法平台 WEKA[27] 中的 3 个分类算法进行对比:朴素贝叶斯(NB)[28],ADTree[26,28],SVM[28].将样本分成 10 份,轮流选择其中一份作为测试样本,其余作为训练样本,计算 4 种算法每次实验的 *TPR* 和 *TNR*,结果见表 2.

由表 2 可知,SVM 可对正常样本和恶意样本实现较好检测,但与 PNN 相比,对恶意脚本的检测准确率无优势且起伏较大,恶意样本集含有多个不同规模与具有层次结构的聚类,可导致 SVM 训练效果降低. NB 可对恶意脚本实现较好检测,但对正常脚本的检测率较低,正常脚本类别特征之间较强的关联性降低分类准确度. ADTree 对正常脚本和恶意脚本的检测结果都不高,ADTree 适用于数据集较小、特征之间关系清晰的情况,对噪声较敏感,并不适合样本特征较多、特征之间关系错综复杂的脚本集.

综上所述,与其他 3 类算法相比,对脚本样本集的非均匀性有较强适应能力的 PNN 更适合脚本特征数据.它使用含有径向基函数的前馈型神经网络,采用多变量 Parzen 窗估计不同类的概率密度函数,使其在聚类分布较复杂的模式分类问题中具有较强的非线性识别能力,可保持非线性算法的高精度,收敛性较好.实验表明其对正常样本和恶意样本的检测率较高,在对比的 4 种算法中总体分类性能 G_{mean} 最好.

表 2 4 种算法的分类性能评价指标值对比

Table 2 Comparison of values of evaluation indicator for classifier performance among 4 algorithms

样本序号	PNN			SVM			NB			ADTree		
	TPR	*TNR*	G_{mean}	*TPR*	*TNR*	G_{mean}	*TPR*	*TNR*	G_{mean}	*TPR*	*TNR*	G_{mean}
1	0.9942	0.92	0.96	0.9884	0.90	0.94	0.8343	0.92	0.91	0.9709	0.74	0.85
2	0.9913	0.90	0.94	0.9884	0.82	0.90	0.8517	0.98	0.91	0.9797	0.66	0.80
3	0.9971	0.90	0.95	0.9942	0.84	0.91	0.8372	0.88	0.86	0.9826	0.62	0.78
4	0.9971	0.92	0.96	0.9826	0.94	0.96	0.9099	0.96	0.93	0.8779	0.94	0.91
5	0.9971	0.98	0.99	0.9913	0.92	0.95	0.8343	0.96	0.89	0.8750	0.92	0.90
6	0.9913	0.94	0.97	0.9855	0.92	0.95	0.8256	0.94	0.88	0.9767	0.66	0.80
7	1.0000	0.86	0.93	0.9971	0.86	0.93	0.8488	0.92	0.88	0.8866	0.84	0.86
8	0.9971	0.92	0.96	0.9942	0.90	0.95	0.8023	0.88	0.84	0.8692	0.90	0.88
9	0.9971	0.96	0.98	0.9855	0.94	0.96	0.7878	0.94	0.91	0.8517	0.90	0.88
10	0.9971	0.88	0.94	1.0000	0.88	0.94	0.8663	0.98	0.92	0.9826	0.72	0.84
均值	0.9959	0.92	0.96	0.9907	0.89	0.94	0.8398	0.94	0.89	0.9253	0.79	0.85

4.2.3 与其他检测工具对比

本节检验 ClamAV[29]、JSAND 及商业杀毒软件金山毒霸、Avast 和微软 MSE 等工具检测恶意脚本代码的能力,不考虑对正常文件的分析.本文分类器在第一个实验中使用 10 折交叉验证方式得到恶意脚本检测率为 91.8%.JSAND 的检测结果为 187 个恶意,127 个可疑,186 个正常,结果为可疑与恶意的脚本都看成恶意脚本.检测结果如表 3 所示.

表 3 中,Avast 检测率(85.6%)最高;JSAND 和微软 MSE 检测率较高;金山毒霸和 ClamAV 检测效

果不好,稍高于 20%. ClamAV 基于特征值难以达到较高检测率;JSAND 是动态执行脚本,不能检测对其蜜罐无影响的恶意脚本;其他工具的技术未公布,无法分析原因. 本文分类器使用更全面科学的脚本特征集,加上有效的 PNN 分类算法,可实现较好的检测效果.

表 3　不同工具对恶意脚本的检测结果

Table 3　Detection results of different tools for malicious script

	JCAD	JSAND	ClamAV	MSE	Avast	金山
TN	459	314	118	327	428	133
TNR	0.918	0.628	0.236	0.654	0.856	0.266

5　结 束 语

本文设计基于多类特征与 PNN 的恶意脚本检测算法,实现检测 JavaScript 恶意脚本的离线分析系统 JCAD. 获取的混淆化特征、统计特征和危险序列特征,既包含总体分析,又包含基于程序语义的内容分析. 危险序列特征获取恶意脚本中频繁使用的操作序列,在实际应用中尤为重要. 系统使用适应恶意脚本集特点的 PNN 分析提取的脚本特征,实现恶意脚本代码的自动检测. 最后通过 3 组实验证明系统能达到较好的检测效果,且训练简单,鲁棒性较好,自组织能力较强,检测速度较快. 在后续工作中,将充分考虑典型恶意行为的表现形式,进一步提高恶意脚本的分析与检测水平.

参 考 文 献

[1] Negrino T, Smith D. JavaScript: Visual QuickStart Guide. 8th Edition. Berkeley, USA: Peachpit Press, 2012

[2] Xu W, Zhang F F, Zhu S C. The Power of Obfuscation Techniques in Malicious JavaScript Code: A Measurement Study // Proc of the 7th International Conference on Malicious and Unwanted Software. Fajardo, USA, 2012: 9-16

[3] Likarish P, Jung E J, Jo I. Obfuscated Malicious JavaScript Detection Using Classification Techniques // Proc of the 4th International Conference on Malicious and Unwanted Software. Montreal, Canada, 2009: 47-54

[4] Egele M, Wurzinger P, Kruegel C, *et al*. Defending Browsers against Drive-by Downloads: Mitigating Heap-Spraying Code Injection Attacks // Proc of the 6th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Como, Italy, 2009: 88-106

[5] Fraiwan M, Al-Salman R, Khasawneh N, *et al*. Analysis and Identification of Malicious JavaScript Code. Information Security Journal: A Global Perspective, 2012, 21(1): 1-11

[6] Blanc G, Ando R, Kadobayashi K. Term-Rewriting Deobfuscation for Static Client-Side Scripting Malware Detection[EB/OL]. [2014-09-28]. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5720649

[7] Blanc G, Miyamoto D, Akiyama M, *et al*. Characterizing Obfuscated JavaScript Using Abstract Syntax Trees: Experimenting with Malicious Scripts // Proc of the 26th International Conference on Advanced Information Networking and Applications Workshops. Fukuoka, Japan, 2012: 344-351

[8] Al-Taharwa I A, Lee H M, Jeng A B, *et al*. RedJsod: A Readable JavaScript Obfuscation Detector Using Semantic-Based Analysis // Proc of the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. Liverpool, UK, 2012: 1370-1375

[9] Lu G, Debray S. Automatic Simplification of Obfuscated JavaScript Code: A Semantics-Based Approach // Proc of the 6th IEEE International Conference on Software Security and Reliability. Gaithersburg, USA, 2012: 31-40

[10] Cova M, Kruegel C, Vigna G. Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code // Proc of the 19th International World Wide Web Conference. Raleigh, USA, 2010: 281-290

[11] Nazario J. PhoneyC: A Virtual Client Honeypot[EB/OL]. [2014-09-28]. https://www.usenix.org/legacy/events/leet09/tech/full_papers/nazario/nazario.pdf

[12] Harstein B. Jsunpack[EB/OL]. [2012-04-08]. http://jsunpack.jeek.org/dec/go

[13] Curtsinger C, Livshits B, Zorn B G, *et al*. ZOZZLE: Fast and Precise In-Browser JavaScript Malware Detection[EB/OL]. [2014-09-28]. http://research.microsoft.com/en-us/um/people/livshits/papers/pdf/usenixsec11b.pdf

[14] Mozilla Foundation. Rhino: JavaScript for Java[EB/OL]. [2012-04-08]. http://www.mozilla.org/rhino

[15] Canali D, Cova M, Vigna G, *et al*. Prophiler: A Fast Filter for the Large-Scale Detection of Malicious Web Pages // Proc of the 20th International World Wide Web Conference. Hyderabad, India, 2011: 197-206

[16] Seifert C, Steenson R. Capture-HPC[EB/OL]. [2012-04-08]. https://projects.honeynet.org/capture-hpc

[17] Dewald A, Holz T, Freiling F C. ADSandbox: Sandboxing Java Script to Fight Malicious Websites[EB/OL]. [2012-04-08]. http://www.iseclab.org/papers/adsandbox-sac10.pdf

[18] Rieck K, Krueger T, Dewald A. Cujo: Efficient Detection and Prevention of Drive-by-Download Attacks // Proc of the 26th Annual Computer Security Applications Conference. Austin, USA, 2010: 31-39

[19] Choi J H, Kim H Y, Choi C, *et al*. Efficient Malicious Code Detection Using *N*-Gram Analysis and SVM // Proc of the 14th International Conference on Network-Based Information Systems. Tirana, Albania, 2011: 618-621

[20] Specht D F. Probabilistic Neural Networks. Neural Networks, 1990, 3(1): 109-118

[21] Moody J, Darken C J. Fast Learning in Networks of Locally-Tuned Processing Units. Neural Computation, 1989, 1(2): 281-294

[22] Yang S Y. Matlab Technology to Realize the Pattern Recognition and Intelligent Computing. Beijing, China: Publishing House of Electronics Industry, 2008 (in Chinese)
(杨淑莹. 模式识别与智能计算——Matlab 技术实现. 北京:电子工业出版社, 2008)

[23] Quinlan J R. C4.5: Programs for Machine Learning. San Mateo, USA: Morgan Kaufmann Publishers, 1993

[24] Lielmanis E. JS Beautifier[EB/OL]. [2012-04-08]. http://jsbeautifier.org

[25] Duda R O, Hart P E, Stork D G. Pattern Classification. 2nd Edition. New York, USA: John Wiley & Sons, 2000

[26] Pandya A S, Macy R B. Pattern Recognition with Neural Networks in C++. Boca Raton, USA: CRC Press, 1996

[27] Kirkby R, Frank E. WEKA[EB/OL]. [2012-04-08]. http://www.cs.waikato.ac.nz/ml/weka/

[28] Han J W, Kamber M, Pei J. Data Mining: Concepts and Techniques. 3rd Edition. Waltham, USA: Morgan Kaufmann Publishers, 2011

[29] Kojm T. Clam AntiVirus[EB/OL]. [2012-04-08]. http://www.clamav.net