面向恶意网页的静态特征体系研究①

刘 畅

(南京理工大学 计算机科学与工程学院, 南京 210094)

摘 要: 恶意网页是一种新型的 Web 攻击手法, 攻击者通常将一段恶意代码嵌入网页中, 当用户访问该网页时, 恶意代码会试图利用浏览器或其插件漏洞在后台隐秘地执行一系列恶意行为. 针对恶意网页静态特征抽取问题, 本文从已有的特征中选取了 14 个信息增益值较高的特征, 并通过分析恶意网页的混淆手法提出了 8 个新的特征, 共同组成了 22 维的静态特征体系. 此外, 针对已有特征抽取流程提出两点改进:对不同编码格式的原始网页进行预处理; 回送 JavaScript 脚本动态生成的的 HTML 代码, 用以进一步抽取 HTML 相关特征. 实验表明, 在不均衡数据集和均衡数据集上, 本文的特征体系具有一定的有效性.

关键词: 恶意网页; 特征抽取; 静态特征体系; 信息增益; JavaScript

Research on Static Feature System in Malicious Web Pages

LIU Chang

(School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China)

Abstract: Malicious Web pages is a new kind of Web-based attack method. In drive-by-download exploits, attackers embed malicious code into a Web page. When a victim visits this page, the code attempts to download and execute malwares by exploiting vulnerabilities in browser or its plugins. Considering the problem of extracting static feature from malicious Web page, this paper selects 14 static features based on information gain theory and proposes 8 new static features are proposed by analyzing obfuscated scripts. In addition, two improvements of original feature extraction process are proposed as follows: preprocessing for original Web page based on different code format; reprocessing HTML code which are dynamically generated by JavaScript to further extract HTML features. The experimental result shows that, on unbalanced data set and balanced data set, our static feature system is provided with a certain validity.

Key words: malicious Web pages; feature extraction; static feature system; information gain; JavaScript

1 引言

随着互联网的快速发展, 黑客的攻击目标逐渐从服务器端转向存在众多安全漏洞的客户端, 越来越多的用户受到病毒或木马等恶意网页的攻击, 造成隐私泄漏、财产损失等问题. 当用户访问被嵌入恶意代码的网站时, 浏览器解析从服务器取回的恶意网页, 恶意网页利用浏览器或其插件漏洞在后台隐蔽地执行恶意代码或下载恶意的可执行文件. 恶意代码的执行并不会影响网页的正常内容显示, 所以用户不会有所觉察.

研究发现, 攻击者对嵌入到网页的恶意代码进行精细的加密和混淆是恶意网页检测的瓶颈. 混淆与加密手法包括对变量内容的加密, 对函数名、变量名的混淆, 对代码逻辑结构的混淆, 代码排版和格式的混乱等. 通过以上手段, 恶意网页能有效避开杀毒软件的检测, 执行恶意行为. Wei Xu 等人在 2012 年检测了市场上最主流的 20 款杀毒软件对混淆的恶意代码的检测能力[1], 平均检测率为 86.85%, 针对某些具体的混淆手法检测率更低. 由于JavaScript的语法灵活性与使用广泛性, 目前大多数恶意脚本都以其作为载体, 本文也以这类恶意网页作为研究对象.

现有的恶意网页检测方法可根据是否执行页面代码分为动态检测和静态检测两种. 动态检测方法会在

① 收稿时间:2015-10-27;收到修改稿时间:2015-12-10 [doi: 10.15888/j.cnki.csa.005257]

蜜罐系统或虚拟环境中完整地执行待检测的网页代码, 通过监测系统变化来判断网页是否带有攻击行为[2]. 动态检测方法有两点不足:①检测环境和攻击脚本发动有效攻击所需环境不匹配, 从而将恶意网页误判为正常网页, 出现漏检; ②需要较高的时间代价等待攻击成功并表现出被感染的特征, 每个页面大约需要消耗两分钟的时间[3]. 静态检测方法主要基于特征匹配, 其优势在于时间消耗低, 不足则体现在三方面:①由于恶意代码的形式变化多端, 已有的静态特征并不能充分刻画一些新型的攻击手法, 从而产生漏检; ②真实网络流量中存在大量良性的混淆网页, 其混淆的目的在于保护代码的版权, 已有的一些静态特征不能有效区分良性和恶性的混淆页面, 因而引起误检; ③现有的静态特征主要是根据经验提出的, 没有通过理论方法证明特征的有效性, 存在一定的不可靠性.

鉴于以上背景, 本文通过分析最新的恶意网页的混淆手法, 主要针对由JavaScript编写的攻击代码来构建恶意网页的静态特征体系. 通过计算已有特征的信息增益值, 从中选取了 14 个信息增益值高的特征. 特征的信息增益值体现了该特征对分类的贡献程度, 信息增益值越大, 该特征对分类越有利. 在此基础上, 我们又提出了8个特征与之前的14个特征共同组成了新的静态特征体系. 实验表明, 针对本文采集的网页样本所构造的均衡和不均衡数据集, 本文提出的特征体系均具有一定的有效性.

2 相关工作

近年来恶意网页检测已成为网络安全领域的重要研究课题. 国内外学者就恶意网页的检测提出了一系列方法, 主要包括动态检测和静态检测两方面.

静态检测方面, Seifert 等人于 2008 年抽取了恶意网页的部分静态特征[5], 是较早的利用静态方法检测恶意网页的学术研究之一. 然而, 这些特征全部来源于恶意网页的 HTML 代码, 并没有对由 JavaScript 编写的恶意脚本进行特征抽取. 实验结果表明, 利用这些特征进行静态检测的误检率为 5.88%, 但漏检率为 46.15%; 同年, 张昊等人将判断矩阵法应用于恶意脚本检测中[4], 该方法仅对利用 encode 和 escape 函数加密的脚本有效, 而利用 fromCharCode、parseInt 等函数配合 eval 函数也能执行加密代码, 此外还有攻击者自定义的各种加密混淆方法, 因此判断矩阵法会产生一定量的漏报; 2010 年, Cova 等人提出了一个名为 Prophiler 的针对大规模恶意网页的静态过滤器[8]. Prophiler提取了19个HTML相关特征、25个JavaScript相关特征, 此外, 还针对恶意站点的 URL 和主机名分别构建了 12 个和 21 个特征. 综合四部分的特征, Prophiler 误检率为 9.88%, 漏检率降到 0.77%; Wei Xu 等人则从混淆的函数定义、伪装的函数调用、混淆的恶意参数三方面抽取了 JavaScript 的静态特征[9].

动态检测方面, Wang 等人[3]使用 BHO(browser helper object, 浏览器辅助对象)捕获浏览器访问被挂马网页时各加载页面之间的链接关系, 并用回溯的方式构建被挂马网页的感染链; PHoneyC[11]在低交互式客户端蜜罐环境中解析页面并用一个独立的脚本引擎执行提取出的脚本, 通过在脚本引擎上下文中模拟出一些已知的漏洞插件, 并结合运行时参数检查来检测恶意脚本; Cova 等人在 2010 年发表的论文中提取了恶意脚本的动态特征[6], 他们利用 HTMLUnit[7]监测脚本的执行, 从页面跳转、脚本混淆、字符串操作、插件函数执行等方面提取了 10 个动态特征, 实验结果表明, 漏检率仅为 0.2%.

3 静态特征体系构建

本节针对近年来几类流行的恶意网页模式, 在相关工作的基础上, 从 HTML 和 JavaScript 两方面提取了 22 个静态特征, 其中包含 6 个与 HTML 相关的特征和 16 个与 JavaScript 相关的特征. 我们抛弃了已有的但信息增益值较低的特征, 保留的特征均有较高的信息增益值, 有利于分类器区分出正常网页和恶意网页.

3.1 静态特征抽取流程

现有的研究对网页进行特征抽取的方法通常是先利用 HTML 解析工具解析网页源代码, 分离出 HTML 代码和JavaScript代码, 再分别检测有没有可疑特征出现. 本文在传统的恶意网页静态特征抽取系统的基础上做了两点改进:

① 原始网页预处理

针对不同地区不同国家或地区的网页, 其编码方式各不相同, 对所有编码进行转码耗时较大. 为了对不同编码的网页进行高效统一的存储和处理, 我们在不影响恶意特征的识别的前提下, 去除了原始网页代码中所有 0x00 字符, 并将非 ASCII 范围内的字符替换为控制字符. 原因是控制字符是不可见的, 通常很少

被使用. 处理后可直接用单字节字符串存储网页代码, 不会出现 0x00 导致的截断问题, 也避免了因非法字符导致转码失败的截断问题.

② 回送动态生成的 HTML 代码

通过对恶意网页的分析, 我们发现有些恶意的 HTML 元素并不是直接写在 HTML 代码中, 而是做为字符串嵌入到 JavaScript 脚本, 再通过 JavaScript 的 document.write 函数动态生成的. 这些恶意元素包括: 含有恶意链接的隐藏<iframe>标签, 包含加密字符串的<p>标签或标签等. 如果不对动态生成 HTML 做解析, 将影响到对网页 HTML 特征的统计, 甚至遗漏一些重要的恶意特征. 因此, 有必要将这些恶意元素从 JavaScript 中解析出来, 回送给 HtmlCxx[10]进行 HTML 部分的特征抽取. 我们对通过三种形式动态生成的 HTML 代码进行还原, 如图 1 所示. 形式 a 中, varName 是字符串变量, 字符串内容即为动态生成的 HTML 代码; 形式 b 中, HTML 代码直接以字符串的形式作为函数参数; 形式 c 较为复杂, HTML 代码由若干个字符串变量和若干个字符串拼接而成.

```
a. document.write( varName );
b. document.write( "<iframe ... /iframe>" );
c. document.write( "<iframe" + ... + varName1 + ... );
```

图 1 动态生成 HTML 代码的三种形式

图 2 所示为改进后的特征抽取流程图, 上述两处改进对应图 2 中第 1 步与第 5 步. 各步骤具体工作如下:

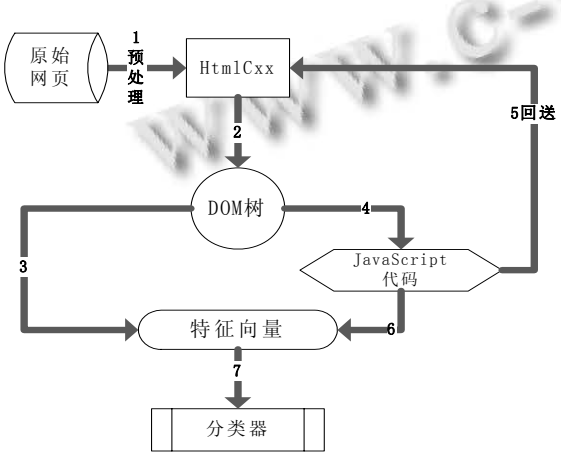

图 2 特征抽取流程

(1) 对原始网页进行预处理;

(2) 利用 HtmlCxx 解析输入的网页源码:HtmlCxx 是一款简洁的 HTML 代码解析器, 能将 HTML 代码解析为一棵 DOM 树, 可用类似 STL 的迭代器遍历 DOM 树来获取 HTML 的标签、属性和内容等信息;

(3) 遍历 DOM 树, 抽取 HTML 相关特征写入特征向量;

(4) 在遍历 DOM 树的过程中提取<script>标签下的 JavaScript 代码;

(5) 还原出 JavaScript 代码中通过 document.write 函数动态生成的 HTML 代码;

(6) 从第⑷步产生的 JavaScript 代码中抽取相关特征写入特征向量;

(7) 利用机器学习方法对特征向量进行分类.

3.2 静态特征集合

Cova[8]提出的 prophiler 过滤器共抽取了 77 个静态特征, 其中包括 33 个关于 Host 和 URL 的特征, 这两类特征不在本文研究范围之内. 我们重点考察了其提出的 19 个 HTML 相关特征和 25 个 JavaScript 相关特征, 通过在 50 组正负样本上计算特征的信息增益选取了信息增益最大的 14 个特征纳入到我们的静态特征集合中. 这 14 个特征包括: (1)页面包含 URL 的数量; (2)内嵌或外链<script>的数量; (3)是否出现<meta>标签; (4)JavaScript 脚本中空白字符占比; (5)每行 JavaScript 脚本的平均长度; (6)JavaScript 脚本占页面所有代码的比例; (7)JavaScript 脚本的总长度; (8)长字符串的数量; (9)最长字符串的长度; (10)字符串的平均熵; (11)字符串的平均长度; (12)隐藏元素的数量; (13)小区域元素数量; (14)字符串修改函数的数量. 这些特征之间具有一定的关联性. (2)、(3)、(12)及(13)号特征与 HTML 代码相关, (12)和(13)号特征可考虑合并为一个特征, 因为很多小区域元素在显示效果上接近于隐藏. (4)、(5)、(6)及(7)号特征均根据页面所有 JavaScript 代码统计得出, 但(4)和(6)号特征侧重描述 JavaScript 代码排版, 而(5)和(7)号特征侧重描述 JavaScript 代码量. (8)、(9)、(10)及(11)号特征均与字符串相关, 前三个刻画了字符串长度, 最后一个则体现了字符串的混乱程度.

我们从采集到的 28701 个正常网页中随机选取了 50 组正常网页, 每组 530 个, 分别与 530 个恶意网页组成 50 组实验数据计算以上特征的信息增益. 这些特

征在 50 组实验数据上的信息增益平均值如表 1 所示.

表 1　prophiler 中信息增益均值最大的 14 个特征

特征编号	信息增益均值	特征编号	信息增益均值
(1)	0.7299	(8)	0.2079
(2)	0.5075	(9)	0.1802
(3)	0.4732	(10)	0.1589
(4)	0.4059	(11)	0.1334
(5)	0.3901	(12)	0.1304
(6)	0.2954	(13)	0.1133
(7)	0.2583	(14)	0.1132

在此基础上, 我们通过跟踪恶意网页样本的执行流程, 设计了十多个特征, 但计算结果表明其中一些特征的信息增益值过低, 实验时发现添加这些特征对系统的质量没有提升作用, 所以没有纳入到最终的特征体系中, 原因可能是这些特征在恶意网页和正常网页中均有出现, 或是和已有特征的作用重叠, 这些特征包括:多个字符串连加、多个函数连加、eval 函数中调用自定义 JavaScript 函数等. 保留的 8 个特征如下:

(1) 页面中所有 HTML 标签数量, 包括已知标签和未知标签. 对从 document.write 函数中还原出的 HTML 代码也进行统计.

(2) JavaScript 动态生成的 HTML 代码是否出现隐藏的 iframe:通过 document.write 函数动态生成的 iframe 很可能包含恶意站点的 URL, 如图 3 所示. 隐藏手法有多种, 常见的包括:元素位置偏离页面可视区域, 元素区域面积过小, 元素设置为透明属性等. 该特征从特征抽取流程的第 5 步获取.

```
var iframe = "<iframe src=\                          恶意站点
"http://culturemerge.ga/AgJVAhoAGFpMUAVU.html\"
width=\"468\" height=\"60\"
style=\"position:absolute;left:-10000px;\"></iframe>";
document.write(iframe);
                                  设置元素样式为隐藏
```

图 3　动态生成隐藏的 iframe

(3) JavaScript 脚本中, 单行包含多条语句的行数占总行数的比例:正常 JavaScript 脚本中通常一行一条语句, 恶意网页中会故意混淆页面排版, 将多条语句写在一行.

(4) JavaScript 脚本中包含正则表达式个数.

(5) 顺序命名的变量:许多恶意网页是通过自动化混淆工具产生的, 混淆后的代码可能出现一系列按顺序命名的变量, 如图 4 所示.

```
var NWZQgJQ2 = '61KjSQH5jGymnuWOKJwGYAoaFphGAgjy-1BkAkW',
NWZQgJQ3 = 'ak1VcVVXN3BDVzluVGhCVjRwN2NNXzJsdGSGpMQ0Qlpa',
NWZQgJQ0 = 'sSi1I0aBH47E756Wel-SPInuMU_IpL1VAcQsepP1YlsdA=',
NWZQgJQ1 = 'ugwpc.bimowamokykpps.net:80',
NWZQgJQ6 = 'OEpDdWl6RTFtY2NDUHpYNXBCempmT1VzaT',
NWZQgJQ7 = 'ugwpc.bimowamokykpps.net:80',
NWZQgJQ4 = 'ugwpc.bimowamokykpps.net:80',
NWZQgJQ5 = '1Q8MmBaKp7fhpiFQ7dXykYosspRAw75_LuzRiAlDyJ_9';
```

图 4　顺序命名的变量

(6) JavaScript 脚本中无规则变量名占所有变量名的比例:正常网页中的变量名长度通常较短并且见名知意, 但恶意网页中的变量名则非常混乱、毫无规则, 包含大小写字母、数字或符号, 这类变量名的熵通常较大. 如图 5 所示.

```
function NYeHRVSaKA ( TRRpuQgFo ) {
  var uVbJfyFe, DnCMnxMWwE = TRRpuQgFo['length'];
  for(uVbJfyFe=0; uVbJfyFe<DnCMnxMWwE; uVbJfyFe++) {
   SpHpepsgZJ[uVbJfyFe]=
      TRRpuQgFo['charCodeAt'](uVbJfyFe)['toString'](16);
  }
  return SpHpepsgZJ['join'](");
}
```

图 5　无规则变量名

(7) 检测浏览器及插件版本信息的 JavaScript 关键字数量, 包括:navigator、appVersion、ActiveXObject 等.

(8) JavaScript 脚本中无规则函数名占所有函数名的比例.

同样, 我们计算了这些新特征的信息增益值均值, 结果记录在表 2 中. 结果显示这些特征的信息增益均值普遍较高, 对分类有较好贡献.

表 2　扩充特征的信息增益均值统计

特征编号	信息增益均值	特征编号	信息增益均值
(1)	0.4996	(5)	0.1805
(2)	0.2180	(6)	0.1392
(3)	0.2142	(7)	0.0941
(4)	0.1833	(8)	0.0928

综上所述, 我们针对 HTML 和 JavaScript 代码, 利用信息增益进行特征选择, 构建了总量为 22 维的静态特征体系. 在实验部分我们将给出该特征体系在不同分类器下的漏检率和误检率.

4　实验

本节通过考察静态特征体系对恶意网页的检测能

力来检验其有效性. 实验使用了 WEKA[12]提供的三种经典的机器学习分类器: 逻辑斯蒂回归、朴素贝叶斯和随机森林, 并采用五倍交叉验证法检验分类效果. 我们重点关注的指标为恶意网页的漏检率和误检率. 漏检率为恶意网页中被判别为正常的网页占所有恶意网页的比例; 误检率为正常网页中被判别为恶意的网页占所有正常网页的比例. 两个指标中, 我们更关心的是漏检率, 因为误检影响的是用户对正常网页的访问, 而漏检则可能导致用户的电脑遭受恶意攻击, 漏检带来的风险远大于误检. 针对本文采集的实验数据, 我们的特征体系与文献[5]和[8]提出的静态特征体系进行了对比实验. 由于正常网页和恶意网页的形态不断变化, 实验结果仅用于证明本文提出的特征体系对于检测最近流行的恶意网页的有效性, 不能对不同特征体系在大规模网络流量环境下的表现给出绝对地评判. 文献[5]提出的 8 个静态特征中包含 6 个 HTML 相关特征和 2 个 JavaScript 相关特征. 需要说明的是, 文献[8]中除了提出 HTML 和 JavaScript 相关的特征外, 还包括 URL 和 Host 相关的特征, 由于我们的特征从 JavaScript 和 HTML 中抽取, 所以对比实验中没有加入 URL 和 Host 相关的特征.

4.1 实验数据设置

我们从 Malware-Traffic-Analyssis[4]收集了 2014 年 2 月至 2015 年 8 月曝光的 530 个恶意网页, 爬取了国内外大型网站上共 28701 个正常网页. 由于正负样本不均衡对特征选择和分类器的训练质量都有影响, 因此我们将正常网页随机等分为 50 组, 每组 530 个正常网页样本, 分别与 530 个异常网页组成了 50 组均衡实验数据, 进行均衡数据集下的特征体系有效性检测. 对于不均衡数据的情况, 我们直接使用 530 个恶意网页和 28701 个正常网页作为实验数据.

4.2 均衡数据集下特征体系的有效性

在 50 组均衡实验数据上, 三套特征体系漏检率的平均值如图 6 所示. 图中横坐标取值 LR、NB 和 RF, 分别代指逻辑斯蒂回归、朴树贝叶斯和随机森林. 实验数据表明, 对于均衡数据集, 使用逻辑斯蒂回归和随机森林作为分类器时三套特征体系的漏检率相当, 均低于 2%. 文献[8]和本文的特征体系在朴树贝叶斯下的漏检率均较高, 其原因可能是 WEKA 的朴素贝叶斯模型会自动对特征的取值做离散化处理, 部分特征处理后效果不佳. 各特征体系误检率的平均值如图 7 所示, 三套特征体系在不同分类器上的误检率均为:本文 < 文献[8] < 文献[5].

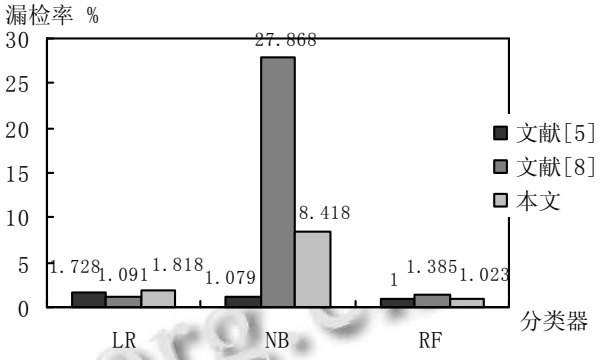

图 6 均衡数据集上不同特征体系的漏检率

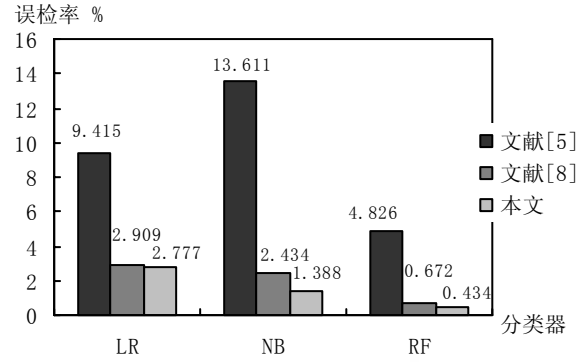

图 7 均衡数据集上不同特征体系的误检率

4.3 不均衡数据集下特征体系的有效性

实验使用的不均衡数据集中, 正常网页的数据量约为恶意网页数据量的 50 倍, 但这更接近实际情况, 因为现实中存在大量正常网页, 但恶意网页相对很少. 因此不均衡数据集上的有效性较均衡数据集上的有效性可能更具说服力.

图 8 表明, 针对不均衡数据集, 本文的特征体系在逻辑斯蒂回归和随机森林上的漏检率要低于其他两套特征体系, 在朴素贝叶斯上的漏检率低于文献[8]给出的特征体系, 略高于文献[5]给出的特征体系. 当采用随机森林作为分类器时, 本文提出特征体系的漏检率为所有漏检率中的最低值, 达到 3.774%. 误检率方面, 图 9 给出的数据表明本文的特征体系在不同分类器上的表现均优于其他两套特征体系. 当采用随机森

林作为分类器时, 本文提出特征体系的误检率仅为0.007%.

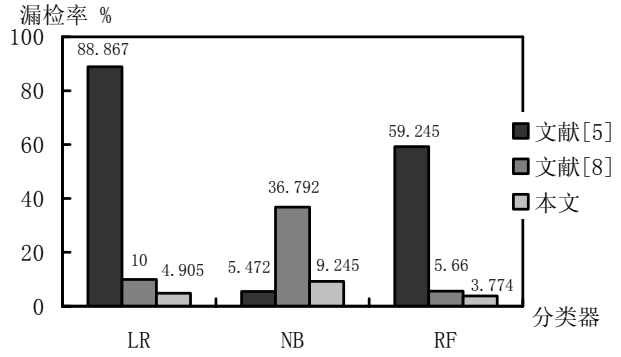

图 8 不均衡数据集上不同特征体系的漏检率

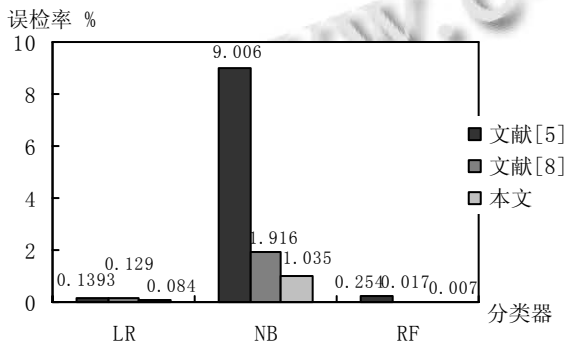

图 9 不均衡数据集上不同特征体系的误检率

5 总结

本文通过改进静态特征抽取流程并提出新的静态特征, 提升了静态特征体系对恶意网页的识别能力. 然而网络安全领域中攻击与防御的博弈是永恒的, 恶意网页的混淆手法变化繁多. 未来的研究中, 我们将继续根据恶意网页混淆加密手法的变化趋势来改进静态特征体系, 尝试挖掘出更具分类能力的静态特征.

参考文献

1 Xu W, Zhang FF, Zhu SC. The power of obfuscation techniques in malicious JavaScript code: A measurement study. Proc. of the 7th International Conference on Malicious and Unwanted Software, 2012, 32(16): 9−16.

2 侯冰楠,俞研,吴家顺.基于预过滤的恶意 JavaScript 脚本检测与分析方法.计算机应用,2015:60−62.

3 Wang YM, Beck D, Jiang XX, Roussev R, Verbowski C, Chen S, King S. Automated Web patrol with strider honeymonkeys: Finding Web sites that exploit browser vulnerabilities. Proc. of the 13th Network and Distributed Systems Security Symp (NDSS), 2006.

4 Zhang H, Tao R, Li ZY, Du H. The application of judgment matrix approach in detection of vicious script in HTML. Acta Armamentarii, 2008, 29(4): 469−473.

5 Seifert C, Welch I, Komisarczuk P. Identification of malicious Web pages with static heuristics. Proc. of the Australasian Telecommunication Networks and Applications Conf. (ATNAC). 2008. 91−96.

6 Cova M, Kruegel C, Vigna G. Detection and analysis of drive-by-download attacks and malicious JavaScript code. Proc. of the 19th Int'l Conf. on World Wide Web (WWW). New York. ACM Press. 2010. 281−290.

7 HtmlUnit. http://htmlunit.sourceforge.net/.

8 Canali D, Cova M, Vigna G, Kruegel C. Prophiler: A fast filter for the large-scale detection of malicious Web pages. Proc. of the 20th Int'l World Wide Web Conf. (WWW). New York. ACM Press. 2011. 197−206.

9 Xu W, Zhang FF, Zhu SC. JStill: Mostly static detection of obfuscated malicious JavaScript code. Proc. of the 3rd ACM conference on Data and application security and privacy. 2013. 117−128.

10 HtmlCxx. http://sourceforge.net/projects/htmlcxx/.

11 Nazario J. PhoneyC: A virtual client honeypot. Proc. of the 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET). Berkeley. USENIX Association. 2009. 6−6.

12 Weka. http://weka.wikispaces.com/.