南京大学

研 究 生 毕 业 论 文

（申请硕士学位）

论 文 题 目　基于支持向量机和节点检测的恶意链接检测系统

作 者 姓 名　刘力铭

学科、专业名称　情报学

研 究 方 向　信息保密与信息安全

指 导 教 师　卢明欣 副教授

二〇一七年五月

学　　号：**MG1414014**

论文答辩日期：**2017** 年 **5** 月　**10**　日

指 导 教 师：　　（签字）

Malicious Link Detection System Based on Support Vector Machines and Node Detection

A Dissertation Submitted to Nanjing University

For the Academic Degree of Master of Management

By

LIU Liming

Supervised by

Associate Prof. LU Mingxin

School of Information Management

Nanjing University

April 2016

南京大学学位论文原创性声明

本人郑重声明：所呈交的学位论文是本人在导师指导下独立进行的研究工作所取得的成果。尽我所知，除了文中特别加以标注引用的内容外，论文中不包含任何其它个人或集体已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中作了明确的说明并表示了谢意。

研究生签名：　　日期：

南京大学学位论文使用授权声明

本学位论文作者同意学校保留并向国家有关部门或机构送交学位论文的复印件和电子文档，可以采用影印、缩印或扫描等复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括刊登）论文的全部或部分内容。论文的公布（包括刊登）授权南京大学研究生院办理。

研究生签名：　　导师签名：　　日期：

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：基于支持向量机和节点检测的恶意链接检测系统

情报学 专业 2014 级硕士生 姓名： 刘力铭

指导教师（姓名、职称）： 卢明欣 副教授

摘要

Web2.0 技术的出现让互联网技术在全球得到了飞速的发展， AJAX 技术让习惯了 Web1.0 时代的静态页面的用户惊喜的发现原来网页也可以动态的改变内容与他们进行交互。在这个背景下，基于 C/S 和 B/S 架构的 Web 应用如雨后春笋一般的出现，极大的丰富了人们的网络生活。但是，互联网发展给用户带来的不仅仅只是各种便利，随之而来的，还有日益严重的网络安全问题，恶意链接是众多网络安全问题中覆盖面最广、也最接近用户的一个。近些年来，移动互联网随着智能手机的迅速普及已经替代了传统的互联网，几乎每时每刻，用户都在通过智能手机中的各种应用链接到移动互联网。传统互联网下的恶意链接问题也同样蔓延到了移动互联网当中，并产生了大量的变种。

因此，如何保证对已知类型的恶意链接意链接样本进行完整识别的同时还能对大量新的恶意链接形式做出迅速的学习和过滤是目前国内外专家学者以及网络安全工作者探讨和研究的重点。目前，国内外针对恶意链接的检测技术，主要有防火墙过滤、利用黑/白名单和特征匹配技术进行检测等，这些方案在多年的网络安全对抗中针对恶意链接取得了不错的识别效果，但是这些方案都是需要通过已知的恶意链接样本指定过滤规则，无法对新出现的恶意链接形式进行识别。然而当前环境下，日益涌现的新技术以及长期的安全攻防已经让黑客有了丰富的攻击经验，因此恶意链接通过代码混淆、链接隐藏等技术发生的变种攻击越来越频发和迅速，传统的检测在解决这些问题方面越来越显得捉襟见肘。面对这一情况，本文在前人研究成果的基础上，提出基于支持向量机和 DOM 检测算法的恶意链接检测系统(SD-MLDS)方案。该系统主要由数据去重模块、黑/白名单检测模块、基于支持向量机的检测模块以及基于 DOM 结构改变的检测几个模块组成。

在数据排重中，通过结合布隆过滤器和改进后的 SimHash 算法，本文提出了兼顾时间和排重效果的 MSHASH-BF 算法。并利用 Map Reduce 对 100 万条数据针对三种算法做了对比试验。

在分类学习检测模块，通过选取分类算法领域中经典的朴素贝叶斯算法、

C4.5、分类回归树以及机器学习领域的支持向量机算法，并根据对恶意链接的研究经验从结构特征、字符特征、账户特征、移动特征等几个方面共选取 36 个属性特征作为训练分类器的特征向量，利用采集的恶意链接测试样本集进行一系列的分类对比试验来证明支持向量机在恶意链接检测方面存在的潜力，并将其用作恶意链接检测系统的核心分类算法。同时，为了及时应对新出现的恶意网络链接，在保持系统开销的前提下提高系统的适用性和识别准确率，本文利用自适应支持向量机在对检测得到或者定期爬取得到的新的恶意链接样本进行增量特征学习来跟新经验分类器，从而不断提高系统的识别能力。

针对网页中以暗链接等形式存在的恶意链接，本文利用 DOM 同层节点比较算法，通过实时监测页面 DOM 结构的变化，发现和提取新增的 DOM 子树，并利用构建的正则表达式对其中的恶意链接进行匹配。将提取出来的恶意链接作为待测样本集利用另外两个模块进行检测。

最后，本文利用实验检测了系统在恶意链接检测方面的效果。虽然本系统方案在恶意链接检测方面针对传统技术方案存在的问题进行了一定的改进，但是没有经过实际的业务检验，因此，其稳定性、系统的开销等还有待改进和提升。同时，网络安全问题并非只是恶意链接这一种，还有恶意脚本、危险舆情等等，如如何扩展和完善本系统方案，使其可以解决更多更复杂的安全问题，是本课题接下来需要继续努力的目标。

关键字：恶意链接检测 数据排重 支持向量机 自适应学习 DOM 文档结构

南京大学研究生毕业论文英文摘要首页用纸

THESIS：Malicious Link Detection System Based on Support Vector Machines and Node Detection

SPECIALIZATION： Information Science

POSTGRADUATE： LIU Liming

MENTOR： Associate Prof. LU Mingxin

ABSTRACT

The emergence of Web2.0 made the Internet been developed and spread rapidly in the world. With the AJAX technology, users were surprised to find that the static web pages they used to browse could change dynamically and interact with them. On this condition, Web applications based on C/S and B/S architecture have sprung up like mushrooms, which enriched people's network life greatly. However, the development of the Internet not only brought convenience, but also the increasingly serious problem of network security, malicious links are the most extensive and the most user closed of these problems. In recent years, with the rapid popularization of smart phones, mobile Internet has replaced the traditional Internet. People can stay on the mobile Internet through various applications in smart phones. Malicious links on the traditional Internet also spread to the mobile Internet with a large number of variants.

Therefore, how to effectively detect malicious links as well as identify new kinds of malicious links is the current explore and research focus of domestic and foreign experts and scholars and network security workers. At present, firewall filtering and utilization of black / white list as well as feature matching are the main detection technology of malicious links, which have achieved good recognition results on the application of network security against malicious links for years, but needs specified sample to make filtering rules. Therefore, these programs are unable to identify the emergence of new forms of malicious links. However, under the current environment condition, the development of new technology and security defense for years have made hackers richer in attack experience, so malicious links attack was more and more frequent and rapid through code obfuscation by the link hiding technology and so on.

On the contrary, traditional detection in solving these problems has become useless. Faced with this situation, this paper proposes a malicious link detection system (SD-MLDS) scheme based on support vector machines and DOM detection algorithm with the help of previous research results. The system is composed of data weighting module, black / white list detection module, detection module based on support vector machine and detection module based on DOM structure change.

In the data weighting, this paper puts forward MSHASH-BF algorithm by combining the SimHash algorithm and the improved bloom filter, which is effective and rapid certificated by compare of 1 million weighting results for three algorithms using

In classification learning detection module, we select classical classification algorithms such as Naive Bayesian and C4.5 and classification and regression tree as well as support vector machine algorithm. We abstract 36 attributes including structural features, character characteristics, account characteristics and moving characteristics as features vector classifier training set for classification. We made a series of comparative experiments to prove the potential of support vector machine in detecting malicious link. At the same time, in order to cope with the emergence of new malicious web links with the promise of applicability and recognition, we choose the adaptive support vector machine to update feature-learning model through new malware samples or links regularly to keep on improving the system the recognition.

We also use a same-layer comparison algorithm of DOM nodes to abstract dark links on the web pages. By real-time monitoring of changes in the structure of DOM page and new DOM subtree extracting, malicious links will be matched with regular expressions. The extracted malicious links are used as samples to be tested for the two other modules.

Finally, this paper tests the effect of the system on malicious link detection. Although the effect of the system in detecting malicious links is improved, it still needs actual business inspection. Therefore, the stability, the cost of the system remains to be improved. At the same time, the network security problem is not just as simple as malicious link, malicious scripts, dangerous public opinion and so on all belong to it. Finding a way to expand and improve the system, so that it can solve more complex security problems is the next stage of this topic.

Keywords: malicious link detection, data row weight, support vector machines, adaptive learning, DOM document structure

目录

图目录

表目录

1 绪论

1.1 课题的研究背景

在过去的十年里，互联网的飞速发展极大的改变了人们的生活。人们可以用互联网完成在线工作、享受在线游戏、接受在线教育等，无时无刻的享受互联网给我们的生活带来的便捷。但是，互联网在极大的丰富和方便了人们的生活、为社会带来了无限的商机和可能的同时，其所蕴含的宝贵财富也招致了黑产行业的势力涌入，从而埋下了许多危机。充斥着互联网的木马、病毒等，给互联网用户带来了极大的安全隐患，严重干扰了正常的网络环境。其突出的一个传播途径，就是恶意链接。黑客可以利用恶意链接进行多种攻击，达到自己的攻击目的。黑客将恶意链接通过各种方式散播出去，只要用户通过这些链接访问了背后的恶意网站，用户的个人隐私信息就会轻易的被泄露而不被用户所察觉，或者恶意的软件就会自动的被安装，进而用户的整个电脑都将被攻击者所控制。

国民经济的持续繁荣离不开互联网的蓬勃发展，2016 年阿里集团总营业额超过三万亿元，相当于我国 GDP 的 4%。其中，天猫和淘宝仅双十一期间的营收就有 1027.49 亿元。在欣欣向荣的背后，巨大的商业利益驱使网络黑客的大量涌入，打造了一条庞大繁杂的地下黑产行业，网络诈骗、网络钓鱼和信息泄露等案例数见不鲜。2016 年 5 月，从著名社交求职网站 LinkedIn 下超 1.67 亿个账户在黑市被公开销售，到谷歌 、雅虎、微软旗下 2.723 亿电子邮箱信息以 1 美元价格流入黑市，层出不穷的网络安全事故让人触目惊心。众多的安全事故凸显出 Web 安全的重要性，利用 Web 漏洞进行攻击是黑客的主要手段。据 360 互联网安全中心发布的数据估计：目前，国内互联网站点存在漏洞的概率为 43.8%，存在高危安全漏洞的概率为 1.3%。

2017 年 2 月，360 互联网安全中心再次发布“2016 年中国互联网安全报告”。报告指出，2016 年该中心共截获各类新增钓鱼网站 196.9 万个，共拦截钓鱼攻击 279.5 亿次。在新增的钓鱼网站中，网站被黑而搭建起来的钓鱼网站为 19.0%，其中 PC 端新增恶意程序样本 1.9 亿个。敲诈者病毒在国内发生两次大规模传播，全国至少有 497 多万台用户电脑遭到了敲诈者病毒攻击。通过对受害者调研，42.6%的受害者不知道感染病毒的原因。预计在 2017 年敲诈者会增长 10 倍，且利用挂马攻击也将再次爆发。据 360 安全中心推算，每年有近 16 亿元的用户财产被钓鱼网站卷走。恶意链接是传播钓鱼网页的最重要渠道，比率高达 52.1%，其它可能的途径如聊天、社交平台，分类资源网络、讨论、微博等。

近年来，随着移动互联网的快速发展，智能化终端的日益流行，服务模式的丰富多样，移动互联网产业正以前所未有的速度增长，应用层出不穷。据统计，2016 年 12 月，我国移动互联网智能终端设备活跃数量达到 9.33 亿，移动互联网用户在所有上网人群中占比已经突破 80%。智能终端设备设备的性能强、易携带等特点使得二维码快速发张成为了新的移动互联网的终端“入口”。QR 码是二维码被广泛应用的典型代表。QR 码由日本 Denso 公司研制，和以前的条形码相比，其读取速度快，存储的信息种类繁多，包括网页链接、保密数据、个人信息等。二维码目前被广泛的应用在社交媒体、网络购物、证件超市等各个领域，它为网民的生活带来了方便的同时，也成为了恶意链接新的携带者和传播者。

和传统的桌面操作系统相比，目前主流的移动端智能操作系统如 IOS、Android 等漏洞相对较多，更容易被黑客关注和攻击。移动设备受限于体积和操作范围等限制，用户在使用移动终端浏览互联网时存在网址输入不便、URL 信息阅读不完整、显示屏幕小无法完整显示 URL 地址等缺陷。此外用户对待移动设备使用时的安全防范意识不强，这些都是导致移动互联网众多安全问题的潜在因素。目前，黑客已经将攻击的范围从传统互联网平台转向了移动互联网和物联网领域，并且其可以利用多种漏洞进行攻击，从而在面对网络安全系统时极大的提高反检测、反侦察能力。

面对如此数量庞大的恶意链接攻击行为，人们迫切的需要一些有效的产品对其进行检测和拦截。现阶段业界广泛使用的安全方案大都建立在黑/白名单库的基础之上，其原理是通过手工或蜜罐对拦截和爬取的 URL 进行鉴别和标记后，将数据按照黑/白名单库分类号存放在数据库中，该方法的优点是对于重复的恶意链接识别率高，但是缺点也很明显，即是用人力成本较高，黑白名单库更新周期较长，对新的变种链接响应不及时等。多年的黑客攻防使得目前的恶意链接数量庞大，攻击方式种类繁多，这些都给给这种防御机制带来了巨大的困扰。因此如何有效而准确的对恶意网页进行实时检测具有深远意义。

1.2 国内外研究现状

目前，在恶意链接检测方面，国内外常用的技术方案主要有以下几种：

（1）特征码匹配技术[7]：该技术的核心思想是对已知的恶意链接的特征进行提取后建立恶意链接特征码库。之后系统会将新的待测链接利用特征码库中的特征码进行匹配，如果待测链接匹配到足够多的恶意链接特征，那么该链接就是恶意链接，系统会对用户发出警告信息提示。由于该技术的实现原理简洁明了，因此特征码匹配技术的在实际的恶意链接检测实践中的可以以较低的系统开销保

证匹配的准确度。但是特征码的提取需要提前准备，所以无法完成实时监测，对于加密和混淆变形后的恶意代码片段更是无能为力。

2.防火墙技术[36]：防火墙是指一个通过软硬件设备组成、在内外网、专用网与公共网之间的建立起来的一个保护屏障，是一种描述网络系统安全性方法的形象说法。防火墙让不同的局域网之间建立起安全网关，从而避免内部网遭受外部非法用户的入侵。服务访问规则、验证工具、包过滤和应用网关是防火墙的主要组成部分。一旦成功建立，该局域网内的计算机所有的出入流量都需要经过此防火墙的过滤和验证。当有外部请求需要通过网关进入内网时，防火墙会根据既有的安全审计规则并对该请求的请求头进行验证，如果经过验证发现该请求存在安全威胁特征，那么就对该请求进行抛弃处理。利用防火墙可以很好的对符合已知规则的恶意链接进行很好的防御，但对经过混淆、变种或者新出现的恶意链接无法进行有效的拦截。

（3）蜜罐检测技术[35]：蜜罐的定义是一个安全资源，它的价值在于被探测、攻击和损害。因此，我们可以把蜜罐类比作一个情报收集系统，通过诱使黑客发动攻击来达到使攻击者暴露攻击的目标、方式和特征等攻击信息的目的，从而帮助安全人员及时了解针对计算机发动的最新的攻击和漏洞，进而有针对性的制定或者更新系统保护方案和检测策略。此外，我们还可以监听攻击者之间的通信，收集收集攻击者的个人信息。蜜罐技术的关键点就是通过仿真环境诱使攻击行为发生，但是，蜜罐的构建和维护需要大量的系统资源，且对于某些无法攻击触发条件的恶意链接无法进行检测。

（4）虚拟机脱壳引擎技术[11]：一旦病毒成功运行，那么用户计算机就已经被感染。为了做到防患于未然，虚拟机脱壳引擎（VUE）技术被提出，该技术的的核心原理和蜜罐技术类似，都是通过一套虚拟仿真环境来诱使攻击者进行攻击，通过追踪攻击行为来获取攻击的相关信息。由于“仿真环境”是与用户的实际操作环境相隔离，因此，即使是通过挂马、病毒链接引入了木马和病毒，其在虚拟环境内的任何攻击操作都不会危害到实际的真实系统环境。同蜜罐技术一样，虚拟机脱壳引擎技术可以有效的检测出恶意链接、木马、病毒等安全问题，但是虚拟环境的搭建和维护消耗的系统资源过大，检测时间过长，且对于某一类长期潜伏在用户计算机的、只在特定的时间出发的病毒同样难以达到较好的检测效果。

（5）机器学习检测技术[13]：该技术的第一步和特征码检测技术有些相似，即分析恶恶意链接后提取恶意链接的特征，利用机器学习算法和大量的样本进行训练学习生成一个模型作为系统的分类器，通过利用该分类器对新的测试链接进行判断和检测。

自机器学习被应用到恶意链接检测以来，许多学者专家提出了多种特征提取

思路。针对 HTML 标签在页面中的使用频率不同，Hou[46]等人利用分层识别的思想将网页按照一定的规则分为不同的层级，在每一层的网页上进行特征提取并按照加权合并，之后利用不同的算法进行分类模型训练和链接预测，通过预测结果来验证不同算法的分类效果。ChitraS[3]等人通过借鉴遗传进化模糊规则技术，利用其和支持向量机并行检测来对恶意链接的特征进行提取。基于机器学习的恶意链接检测相比传统的检测方法在检测效果和应对变种攻击方便提升明显，但机器学习存在训练和学习过程太过复杂，生成的决策规则效率较低等问题。

1.3 课题的研究意义

经过上文的分析可以看出，这些技术的核心是针对已知的恶意链接建立特征码库后，通过模式匹配、字符串匹配等方法对恶意链接进行检测，样本特征的提取好坏直接影响这些方法的检测效果。但是，当恶意链接为了实现反检测而通过编码混淆、链接隐藏等技术进行形式演变时，以上技术无法对这些新的恶意链接形式进行有效的分类和识别。针对这一问题，本文在借鉴以上研究成果的基础上，提出基于支持向量机算法[1]（Support Vector Machine，SVM）和文档对象模型[5]（Document Object Model，DOM）节点改变检测算法的恶意链接检测系统(SD-MLDS)方案。

该方案首先通过数据排重模块利用 MSHASH-BF 算法对数据进行去重处理，处理后的数据先经过黑/白名单检测模块进行匹配检测，将已知的特定类型的恶意链接剔除。之后，将剩下的数据通过分类检测模块检测，同时，为了有效应对恶意链接形式演变过快的问题，该方案利用新检测出的恶意链接样本通过自适应机器学习算法对分类器进行增量更新训练。同时，针对通过懒加载、延迟加载等技术隐藏导致系统无法直接检测的暗链接等恶意链接，该方案通过 DSCD 算法实时监测页面 DOM 结构的跟新，抽取出新增的 DOM 节点，并通过内建的暗链接匹配正则表达式提取恶意链接。本文的研究重点是如何利用自适应支持向量机和 DOM 节点同层比较算法来应对常规检测系统难以识别的恶意链接，达到提高系统的检测准确性和针对变种攻击的应急反应能力，这对于打击黑客的攻击行为、营造安全的网络环境和构建健全的网络安全防御系统具有一定的参考价值。

1.4 论文的结构概述

本文共分为 6 章来阐述，每一章的具体内容概括如下：

第 1 章介绍了当前网络环境中存在的日益严峻的恶意链接问题这一研究背

景，并详细的列举了当前阶段国内外学者和网络安全专家针对恶意链接问题的研究现状和技术进展，总结了这些技术方案的优点和不足，并提出了本文的研究目的和研究价值。

第 2 章介绍了恶意链接和恶意脚本的攻击过程以及 JavaScript 语言的基本知识。该部分针对 JavaScript 的特性、运行原理、页面加载方式、安全性以及恶意链接如何通过 JavaScript 进行攻击进行了详细的介绍。

第 3 章针对朴素贝叶斯算法、ID3、C4.5、分类回归树这几个经典分类算法以及在本系统采用的核心算法之一支持向量机算法进行了详细的介绍。

第 4 章详细介绍了 SD-MLDS 系统的数据排重、黑/白名单过滤、基于 SVM 的恶意链接检测和基于 DOM 结构改变检测的恶意链接提取模块及系统的整体工作流程，并着重介绍了基于布隆过滤器和改进 SimHash 算法的 MSHASH-BF 算法以及用以监测 DOM 结构变化的 DSCD 方案。

第 5 章设计了系统的评价指标，通过详细的实验对系统的几个功能模块进行对比验证。

第 6 章对课题所做的工作进行了总结，同时分析了系统方案中存在的不足之处，对未来的进一步的研究工作进行展望。

2 恶意链接相关技术介绍

本章节对恶意链接和脚本的定义进行了阐述，同时详细分析了作为恶意链接主要攻击手段之一的 Javascript 脚本是如何注入到攻击页面从而达到传播和攻击目的的。此外，本章还会简要概括恶意链接和恶意脚本的常见特征。

2.1 恶意链接和恶意脚本简介

1、恶意链接

恶意链接[22]是一个集合名词，指故意诱导用户访问夹杂恶意脚本从而传播病毒、蠕虫和特洛伊木马的网站的 URL。这类网站通常都有一个共同特点，他们通常情况下是以某种网页形式可以让人们正常浏览页面内容，同时非法获取电脑里面的的各种数据。浏览器作为用户浏览互联网的主要工具，由于开发人员设计和开发软件时的疏漏和对网络安全考虑步骤，会在浏览器及其相关插件中留下漏洞隐患。同时，站长建站时也会因为同样的问题在网站的后台或者数据库的设计开发中留下漏洞。如果没有被及时的通过后续版本或者补丁修复，这些漏洞就会被黑客发现和利用，通过编写包含恶意的 JavaScript 脚本、Applet 应用程序、包含恶意 ActionScript 的 Flash 动画以及 IE 浏览器下常用的 ActiveX 控件等的页面，并将这些页面利用混淆、加密等技术进行伪装使其要么看起来像正常的链接要么无法简单的一眼识别后利用邮件、广告、海报等方式进行传播，一旦用户被欺骗后点击这些链接进入了恶意网页，蠕虫、木马等恶意软件就会在用户毫无察觉的情况下被下载运行，进而控制用户的电脑，盗取用户隐私，破坏用户数据，更严重的甚至会将用户的电脑控制为僵尸机器或者是“肉鸡”从而更快速的传播自己。

2、恶意脚本

网页恶意代码[11]是注入到网页中通过利用系统或者后台漏洞达到攻击目的的代码片段，这些代码的编写大多是利用在前端广泛使用的 JavaScript 语言完成的，并且经过一定的编码和代码混淆技术伪装来逃避系统的判别。因此，利用这些语言编写的恶意攻击代码片断叫做恶意脚本，包含恶意脚本的网页称为恶意页面。通常来说，恶意脚本的攻击氛围直接攻击和间接攻击两种。

（1）直接攻击：是指恶意代码直接包含在恶意网页里面，用户通过恶意链接进入含有恶意脚本的页面后，恶意脚本被运行，从而用户发起包括利用钓鱼对主页进行劫持、窃取用户的个人数据和隐私信息、伪造用户网银身份进行资金诈骗、向网站后台注入非法 SQL 语句夺得站点控制权等恶意攻击。

（2）间接攻击：是指恶意脚本通过一定的方式注入到正常页面，用户访问正常页面后恶意脚本将用户导向伪装的恶意页面实现真正的攻击。常见的挂载手段是利用基于 AJAX 的懒加载技术[38]通过 iframe 标签动态加载。黑客会通过许多网页漏洞扫描工具批量扫描互联网上的页面对应的站点是否存在可以利用的漏洞，一旦发现漏洞就将相应的恶意脚本注入进去。这样，当用户通过正常途径访问这些页面引发恶意脚本加载和执行后，钓鱼链接会通过弹窗或者覆盖页面的方式出现，从而诱导或者强制用户跳转到真正的恶意站点，进而执行黑客指定的攻击行为,这就间接攻击完整的攻击过程。

恶意链接的种类和形式多种多样,但是通过上文的分析可以知道，恶意链接的攻击途径只有直接和间接两种。因此，本文以此为切入点，主要从以下两个方面进行研究：

（1）通过特征提取和匹配来判断一个 URL 是否是恶意链接从而防御直接攻击。

（2）通过网页源码分析和 DOM 结构解析等方式检测页面是否含有暗链、恶意链接和脚本。其中，暗链指存在于页面源代码中而难以被检测系统检测到的代码片段，其在网页源码中的具体形式和特征将在后面章节介绍。

2.2 恶意链接的攻击过程

一般来讲，黑客利用恶意链接攻击用户一般过程为利用漏洞扫描工具扫描在线站点寻找漏洞，在找到可以利用的页面漏洞后，将恶意脚本注入到漏洞网页中，当用户浏览该网页时将会触发恶意脚本的执行，通过重定向、页面劫持等方式诱导用户通过恶意链接进入真正的攻击页面。下图是恶意链接攻击的具体过程：

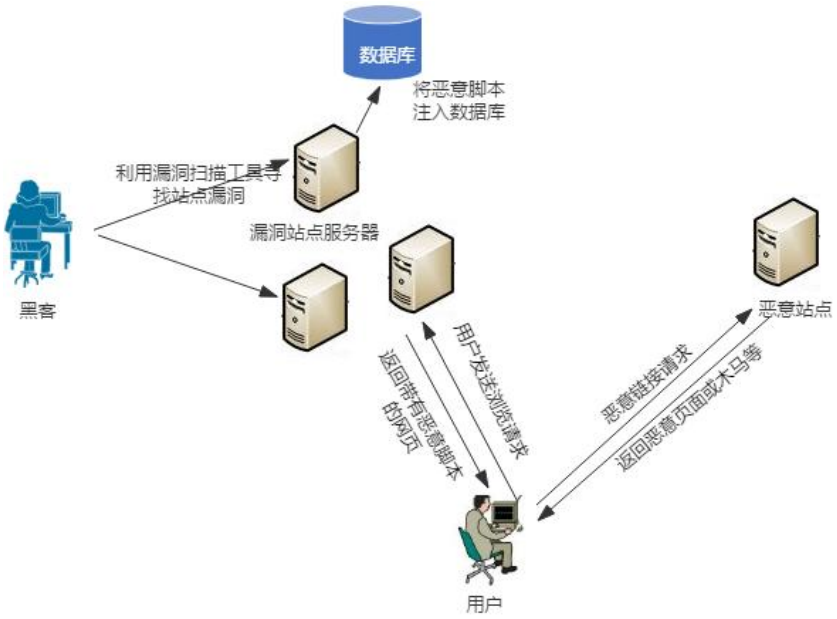

图 2-1 恶意链接攻击的常见过程

通过分析流程图可以将恶意链接攻击的具体步骤总结为一下几步：

Step 1：黑客向漏洞页面注入恶意脚本；

Step 2：用户通过浏览器或者内嵌 WebView 引擎的客户端发送浏览被感染页面的请求；

Step 3：站点的后端服务器处理页面请求后将感染页面发送给客户端；

Step 4：页面在客户端被解析渲染后触发嵌入的恶意代码；

Step 5：执行后的脚本通过弹窗、页面覆盖等方式诱导用户通过恶意链接进入恶意站点；

Step 6：恶意站点开始进行真正的攻击操作，如伪造用户身份进行在线交易、强制安装木马软件、控制用户机器进行自我传播等。

由第六步可知，恶意链接具有诱导性、快速传播性、伪装性等特点，对用户和互联网的环境危害性巨大。相比于过去的木马、蠕虫等攻击主动性攻击模式，即黑客扫描用户计算机本身的漏洞然后利用此漏洞感染用户计算机，恶意链接的攻击在主动性攻击的基础上进一步演变，加入了被动性攻击模式，即先建设包含恶意脚本的伪装页面，之后将该页面的 URL 通过链接加密、链接伪装、脚本注入、重定向等各种方式诱骗用户浏览恶意网页从而达到攻击的目的。由于用户的初次访问请求是处在防火墙、杀毒软件等白名单中的正常站点，因此这些流量数据包能够顺利的通过防火墙的规则验证和过滤，具有很高的攻击成功概率和攻击隐蔽性。

2.3 恶意脚本概述

从之前的章节可以发现，恶意链接的攻击能够顺利完成，需要多方面的条件具备，但是都离不开恶意脚本的注入和执行，目前的前端页面脚本主要是利用 JavaScript 语言进行编写，并在浏览器内嵌的解析引擎解析和执行，接下来本文将对 JavaScript 语言进行相关的介绍。

2.3.1 JavaScript 简介

JavaScript[11]是一种非编译式脚本语言，是在 1995 年，网景公司为了给静态的网页添加交互功能，交由 Brendan Eich 设计和开发出来的脚本语言，并在其旗下的 2.0 版本浏览器中首次使用。JavaScript 和 Java 语言有很大的区别，它无需预编译、通过原型来实现面向对象且对数据类型不敏感。通过 WebView 引擎中内置的 JavaScript 引擎解析执行，它的主要设计目的是应用在 HTML 语言编写的网页中来增加网页和用户的交互功能。目前大部分浏览器引擎是基于 ECMA-262 的第五版标准来实现解析 JavaScript 语言的功能。

作为一个广泛使用的语言，JavaScript 脚本语言具有许多独特的特点：

（1）非编译性：不同于 C++、Java 等需要先编译后执行的编译性语言，JavaScript 是在程序的运行时有引擎对代码逐行解释，是一种解释型的语言。

（2）面向对象：虽然 JavaScript 是解释语言，但是它从设计之初就是面向对象的。他可以实现面向对象语言的全部特性，不同的是 JavaScript 是基于原型来实现面向对象的。

（3）易用性：由于 JavaScript 是弱类型的语言，没有对变量类型做严格的类型区分和限定，因此，相比于 C++和 Java 等强数据类型语言，JavaScript 语言编写程序时较为方便。

（4）动态性：JavaScript 的核心是事件驱动机制，其执行不必经过服务器的解析和帮助，只要解析引擎按照 ECMA 标准设计实现相应的事件接口，那么 JavaScript 程序可以响应鼠标移动、屏幕滚动等事件。

（5）跨平台性：JavaScript 的设计借鉴了适配器模式的设计思想，其执行不限定于用户的操作系统，只要用户执行 JavaScript 程序的环境包含其解析引擎即可。因此一个 JavaScript 脚本在编写后可以在多种端设备上执行

与其它脚本语言相比，JavaScript 的设计初衷就是增加网页同用户的交互，因此，它是在浏览器等端应用上运行的。因此最初程序员们喜欢将许多逻辑用 JavaScript 在端实现从而达到对服务器减去负载的目的，但与此同时也引入了安

全性这一潜在的问题。虽然 在执行时可以将 JavaScript 函数反编译为源码，但在实际应用中，经过代码混淆、代码压缩等处理过后的 JavaScript 代码通常难以分析检测，这正是许多网页能够成功通过安全检测的原因所在。

2.3.2 JavaScript 的解析引擎

JavaScript 的种种优势使得其目前成为主流浏览器的支持实现标准，但是，一直以来，受限于其解释性语言的特性和客户端性能的限制，性能问题一直是主流浏览器厂商的研究重点。2008 年 9 月 2 日，谷歌发布 V8 引擎，极大的提升了浏览器解析 JavaScript 语言的执行渲染速度和稳定性，从此 JavaScript 开始飞速的普及和发展。目前，主流浏览器的引擎除了 V8 以外，还有 Opera 的 Carakan 引擎、Firefox 的 Trace Monkey 引擎、Safari 的 Squirrel Fish Extreme 引擎等。

2.3.3 页面加载 JavaScript 代码的方式

目前，在网页中引入 JavaScript 脚本的方式主要有以下几种：

（1）内嵌式

通过在页面上引入<script>标签将需要的 JavaScript 代码写在该标签内，这是最常用的一种加载 JavaScript 代码的方式，示例代码如下：

```
<script type="text/javascript">
    window.alert('You have been attacked!')
</script>
```

图 2-2 内联式 JavaScript 代码片段

（2）通过<script>标签的 src 属性引入封装的 JavaScript 脚本

在实际的网页开发中，为了实现代码的解耦和组件的复用，开发者们可以将常用的 JavaScript 脚本片段封装成文件， 并在<script>标签的 src 属性中通过相对路径或者 URL 的形式链接外部脚本。示例如下：

```
<script src="/vender/js/jquery.js ></script>
```

图 2-3 利用外链加载封装的 JavaScript 脚本

这样一来将 ，前端可以通过这样的形式来更好的组织开发过程、搭建程序架构，将站点的逻辑层和视图层解耦，方便开发和维护。同时，前端开发人员可以搭建自己的组件库实现代码复用、提高开发效率，有效的减少网页文本大小提升用户浏览体验。

（3）内联式

在 ECMA 标准中，有一种 DOM 0 级事件，可以在 HTML 标签内通过一些内置的事件属性直接添加对应的 JavaScript 代码。例如，针对用户的鼠标点击事件，我们可以在 div 标签内的 onblur 属性引入，并将对应的 JavaScript 代码绑定进去。当用户的鼠标从该 div 移开导致其失焦时，onblur 绑定的 JavaScript 脚本就会被触发，示例代码如下：

```
<div id="example" onclick="window.alert('You have been attacked!')"></div>
```

图 2-4 内联式 JavaScript 代码片段

2.3.4 JavaScript 存在的安全问题

由于 JavaScript 执行时对应的解析引擎无法区分代码的具体行为，因此，黑客注入的恶意脚本同样会被完整的解析运行，因此，我们自然就希望可以通过一些方式控制 JavaScript 脚本的行为。目前，主流的浏览器实现的解析引擎主要通过以下几种方式来实现对 JavaScript 脚本运行时的控制。

（1）无法操作用户的计算机：浏览器将 JavaScript 的操作权限限定在浏览器内部，无法通过 JS 脚本直接删除或者修改用户计算机上的本地文件；

（2）同源策略[6]：虽然 JS 脚本无法直接操作用户的常规文件，但是由于 HTTP 协议的无状态性，服务器需要通过一些措施来识别每一个登录到服务器的用户并追踪其操作，因此无法避免的需要利用 Cookie 等文件来与前端页面的 JS 脚本进行交互，但是正是这个原因导致 JavaScript 语言可以读取和修改 Cookie 文件信息，如果不对不同的 JS 脚本进行区分，那么黑客注入的恶意脚本就会盗取用户身份信息进行恶意操作。因此，浏览器针对这一问题提出了同源策略来限制脚本的跨域访问，使攻击者无法轻易的实现跨区请求伪造（CSRF）[21]或者跨站脚本攻击（XSS）[39]。

此外，许多公司和开发者从实际的业务角度出发会给自己的 JavaScript 脚本设置一些签名。只有脚本通过了签名验证后，用户才可以继续使用浏览器。

2.3.5 JavaScript 常用的攻击技术

近些年随着 Web 技术的快速迭代更新，JavaScript 的应用范围越来越广，已经成为了前端开发人员完成页面逻辑的首要语言，这就使得网络上利用 JavaScript 进行攻击的行为爆发式增长。据 IBMX-Force 2016 报告统计，有 600 个知名网站存在可以被 JavaScript 利用的攻击漏洞，具体包括 Cookie 信息盗取和伪

造、跨域请求伪造、页面重定向以及跨站脚本攻击等等，本文接下来简要的介绍一下 JavaScript 攻击常用的漏洞。

（1）XSS 注入式攻击

跨站脚本[39]（Cross-Site Scripting，XSS）是 JavaScript 常见的一种攻击形式。攻击者将首先检测站点存在的可利用漏洞，然后尝试在页面的输入框内提交特殊的用户文本，多为变形后的 JavaScript 代码，文本在提交到站点服务器后，当普通用户浏览站点时，攻击者注入的恶意代码就会被加载到用户浏览的网页上执行，达到攻击目的。XSS 主要有反射型 XSS、存储型 XSS、基于 DOM 的 XSS 三种。

（2）URL 重定向

URL 重定向[40]（URL Redirection）指的是用户点击链接进入目标页面后，又迅速被强制跳转进入其它的页面的技术。攻击者经常利用该漏洞来进行网络钓鱼、发送匿名邮件等操作来让用户在毫无察觉和准备的情况下从原有的页面跳转到攻击者指定的恶意页面，进而实现攻击者盗取用户隐私、对用户计算机挂马等攻击目的。图 2-5 为是 URL 重定向的的恶意脚本示例。

```
var sData = document.location.search.substring(1),
    sPos = sData.indexOf('url')+4,
    ePos = sData.indexOf('&',sPos),
    newUrl = null;
    if(ePos<0){
        newUrl = sData.substring(ePos);
    }else{
        newUrl = sData.substring(sPos,ePos);
    }
    window.location.href = newUrl;
```

图 2-5 实现 URL 重定向的 JavaScript 代码片段

最终的重定向跳转通常利用 window.location、document.location、document.referrer 等 API 来实现。例如用户点击以下链接：https://baidu.com/redirect.jsp?url=http://www.attacks.com 的目的是访问百度首页，但是该请求的实际目标页面是 http://www.attacks.com 对应的页面，请求将会被自动转发到参数 url 后面对应的恶意网址。更严重的是，攻击者将该链接利用 base64 等编码技术进行编码混淆后的隐蔽程度非常高，难以被检测，普通的用户极易上当受骗。

（3）跨域请求伪造

跨域请求伪造[21]（Cross Site Request Forgery，简称 CSRF），表面上看和 XSS 攻击类似，实际上，XSS 是向网站注入脚本后在用户浏览网页时合法盗取用户的隐私信息，CSRF 恰恰相反，它是通过一定的手段，伪造用户的个人信息来进行攻击，本质上是一种劫持攻击技术。

针对这一问题，主流的浏览器厂商目前一致推行同源策略（Same Origin Policy）来预防 CSRF 攻击。所谓同源，是指域名、协议、端口相同，限制的是 JavaScript 脚本访问非同源页面 Cookie 文件和发送 XMLHttpRequest 请求的行为。例如针对 www.normal.com 和 www.attacks.com 两个页面，origin 页面的 JavaScript 代码在访问用户 Cookie 和发送 Ajax 请求时，如请求 www.attacks.com/malicious/attack.js 时，浏览器会检查该请求的资源是否属于 origin 域，即是否同源，只有同源，这些操作才被允许执行，否则，浏览器会在控制台中报告异常信息，提示“XMLHttpRequest cannot load ‘http://www.attacks.com/malicious/attack.js’. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'null' is therefore not allowed access.”。

根据 2016 年 Standards 报告显示，自 AJAX 技术得到普及一来，网页面普遍利用 AJAX 配合 JSON 数据格式来实现动态页面的效果，因此，CSRF 攻击行为发生的情况和场景非常普遍。攻击者尤其喜欢盗取页面的 JSON 数据为己用，来帮助自己完成恶意攻击。具体的攻击过程如图 2-6 所示。

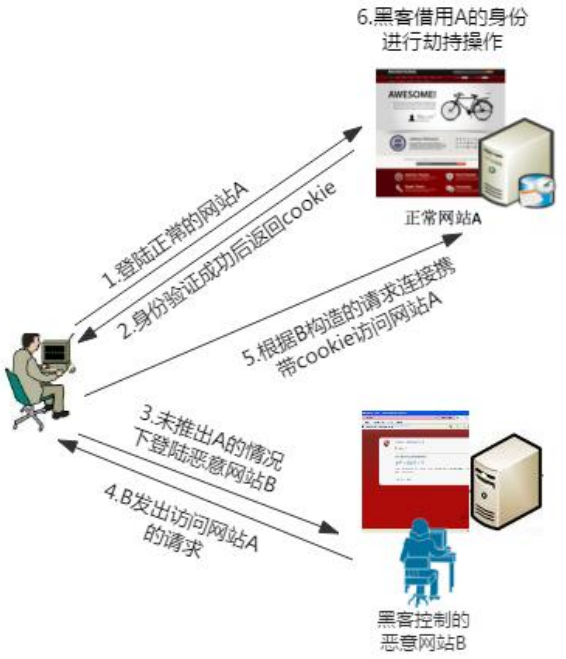

图 2-6 CSRF 攻击过程

假设用户需要访问的目标网页是 http://www. normal.com，那么利用恶意脚本进行 CSRF 攻击的具体过程则分为以下几步：

Step1：首先，用户在浏览器中输入 http://www. normal.com 进入该站点。

Step2：用户打开登陆框，在 form 表单内填写账户名、密码等身份信息后提交表单。

Step3：服务器根据用户填写的信息来对用户进行身份校验，如果用户信息

正确，则返回登陆后的页面，并将相关信息随同 cookie 一起返回给浏览器。

Step4：之后，用户在保持 http://www. normal.com 页面登陆状态的情况下访问了含有恶意脚本的网站 http://www.attack.com。

Step5：恶意脚本盗取用户在 http://www. normal.com 网站的 Cookie 信息后模拟用户交易链接：

http://www.normal.com/transfer_account?useraccount=username&userkey=uerkey&accepter_account_name=attacker&money_amount=10000，并通过 AJAX 向 http://www. normal.com 网站提出交易请求。

（5）http://www. normal.com 网站接受请求验证用户信息通过后将用户账户内的 10000 元现金转入到攻击者的账户中去，这一切的过程都是在用户毫无察觉的情况下发生的。

JavaScript 脚本代码如图 2-7 所示。进而实现了 JavaScript 脚本劫持技术。

```
Object.prototype._defineSetter_('money', function(obj){
    var obj = obj || null;
    for (var fld in this){
        obj += fld+":"+this[fld]+",";
    }
    req.open("GET", "http://www.attacks.com?obj="+escape(obj), true);
    req.send(null);
});
```

图 2-7 JavaScript 劫持攻击的恶意脚本代码

（4）利用二维码的链接隐藏攻击

二维条码/二维码[41]（2-dimensional bar code）是一种新的信息记录方式，它利用许多表示 0 和 1 的黑色和白色图形组成一个大的集合图形来达到表示一定的信息的目的，当需要读取其存储的信息时，可以将二维码利用光电扫描设备扫描处理后得到。

QR 码[18]（Quick Response Code）是二维条码的一种，，其内容可以被移动设备等快速被解码，是针对移动设备设计的一种快捷信息存取方案。因此，QR 码具有二维码的一些基础特性：利用不同的字符集来进行适配不同的编码方案、可以对存储的信息进行完整性检验、可以被相关的读取工具自动处理等。和传统的条形码相比，QR 码具有存储信息更多，识别更加快捷简单邓特点，因此在移动互联网时代得到了迅速的普及和应用。

正是在这样的背景下，黑客将传统的恶意链接借助 QR 码进行隐藏变换从而更方便的在移动设备上用户进行钓鱼、盗取信息等操作。由于 QR 码具有解析快肉眼难以识别和移动智能设备存在屏幕小、输入不便等缺陷，基于 QR 码的攻击隐蔽性极强、危害极大。基于 QR 码的恶意链接攻击具体分为以下几个阶段：

图 2-8 基于 QR 码的网络钓鱼攻击方式

1、攻击者制作伪造页面并将页面的恶意链接通过二维码生成工具生成 QR 码。

2、攻击者将正常的 QR 码通过含有恶意链接的 QR 码替换掉，从而篡改目标网页的 URL 信息。

3、将 QR 码通过微信公众号、网页海报等方式散发出去，将用户引诱至一个假冒登录页面。

4、用户使用解码设备扫描 QR 码，解析出来的链接经过内嵌的 WebView 引擎的 APP 访问，从而达到了攻击者的攻击目的。

2.5 本章小结

本章首先介绍了什么是恶意链接和恶意链接的一般攻击过程，并针对恶意链接实现攻击的主要技术——JavaScript 从语言、解析引擎、页面引入方式、利用漏洞进行攻击的攻击方式做了具体的介绍和阐述。通过本章节和上一章节的介绍，我们已经对恶意链接有了整体的认识，同时在如今攻击手段的多样化、针对防范措施的快速演变性、攻击场景的复杂化的背景下，传统的基于特征匹配、基于黑白名单过滤等恶意链接检测方法已经无法满足当前的安全需求。因此，开发出一个快速、准确、敏感、主动学习反馈的恶意链接检测系统的任务迫在眉睫。由于分类算法和机器学习技术在近几年取得了不小的突破，并在多个领域被应用和实践，效果显著，自然地，网络安全工作者开始尝试将这些技术引入到 Web 安全领域。接下来，本文将介绍自动分类和机器学习中常用的一些技术和算法。

3 分类算法介绍

分类学习算法的核心是通过对已知样本集提取的属性特征进行分析和学习后利用生成的分类模型对未知样本集中的数据进行分类和判断，其过程主要分三步，第一步是提取已知样本集的属性特征组成特征向量，第二步是利用特征向量训练分类模型，第三步是利用生成的模型进行预测分析。以上步骤可以进一步总结为两个状态，学习状态和应用状态。学习状态关注的是如何选取合适的特征向量集进行分类模型训练，应用状态关注的是生成的分类模型对新数据预测的结果，也就是准确率和召回率，并根据结果的好坏对分类模型做进一步的更新优化。目前，分类学习已经在语义分析、图像识别等领域得到了广泛的应用并取得了显著的效果，然而，分类学习算法在恶意链接识别领域尚未得到充分的利用。为了更好的了解分类学习，挖掘其在恶意链接领域的潜力，本章节选取了朴素贝叶斯算法、ID3 算法、C4.5 算法、分类回归树（CART）算法以及支持向量机（SVM）算法这五种经典的分类学习算法，详细分析了这些算法的基本定义、核心思想以及各自的优缺点。

3.1 朴素贝叶斯算法

朴素贝叶斯算法[43]是基于贝叶斯定理与特征条件独立假设的分类方法，其核心是根据贝叶斯定理来对一个未知样本属于某个类别的概率进行预测分析，找到概率最大的类别后将未知样本进行最合适的归类。贝叶斯算法又分为朴素贝叶斯和贝叶斯算网络，前者认为数据的各属性条件独立不相关联，后者则认为实际的决策环境下数据的属性存在一定的相关性。由于朴素贝叶斯算法具有分类过程简单、原理通俗易懂、稳定性比较高、模型训练快分类迅速等优点，本文选择该算法进行具体的分析介绍。朴素贝叶斯算法的具体原理如下：

对于一个未知样本，假定其有 n 个属性 X，将这些属性作为一个向量集合可以得到 $X=(x_1, x_2, x_3, \ldots, x_n)$，同时，假定训练集样本已经分类出了 m 个不同的类别 C，我们用 $C_1, C_2, C_3, \ldots, C_m$ 来表示。然后，通过朴素贝叶斯分类算法进行计算后，我们可以得到样本 X 与不同类别的相似程度，可以将其称为后验概率，将得相似程度按照高低大小排序后，将 X 分配到具有最高结果的类别 C 中，即当且仅当$P(C_i|X) > P(C_j|X)$，$1 \leqslant j \leqslant m$ 且 $j \neq i$ 时，样本 X 就被归类到类别 C_i。

已知贝叶斯定理：

$$P(C_i|X) = \frac{P(X|C_i) * P(C_i)}{P(X)} \quad (3\text{-}1)$$

对于分母$P(X)$，由于训练集的数量在模型建立时就已经确定，因此其大小是个固定值，故而求解最大后验概率时我们只需求解出最大的分子就可以了。此外，如果我们假定训练集样本总量为 S，S_i 为训练集中的样本属于 C_i 的数量，那么我们可以根据公式$P(C_i) = \frac{S_i}{S}$来计算类别 C_i 的先验概率。

通过公式我们可以发现一个问题，就是当训练集和的$P(X)$和分类的类别 C 数量过多时，那么得到$P(X|C_i)$的所耗费的运算时间和性能消耗就会很大。为了改进这一问题，朴素贝叶斯算法提出了一个重要假设，即假定样本的类别 C_i 间彼此独立，这样一来，$P(X|C_i)$的求解公式就变成了式 3-2：

$$P(X|C_i) = \prod_{k=1}^{n} P(X_k|C_i) \quad (3\text{-}2)$$

其中$P(X_k|C_i)$，k=1，2，..，n。

需要注意的是，在实际应用中，朴素贝叶斯算法计算未知样本的后验概率时经常会得到多个最高结果，这表明未知样本的属性类别特征不明显，针对这种情况，可以通过有选择性的调整特征之间的区别程度或者提取差异度更大的属性特征来进行改善。

3.2 决策树算法

决策树[4]（Decision Tree）算法是一种逼近离散函数值的方法，最早可以追溯到 19 世纪 60 年代。先是 J Ross Quinlan 提出了 ID3 算法，该算法减少了决策树的深度，代价是减少对叶子节点的考虑。之后的 C4.5 算法针对 ID3 的问题预测变量的缺值处理、剪枝技术、派生规则等方面做了较大既适合于分类问题，又适合于回归问题。Hunt. E. B 等人于 1966 年提出了概念学习系统 [44](Concept Learning System，CLS)并得到了较大的普及和应用，现在流行的决策树算法基本是对 CLS 的提升和改进。

CLS 的核心原理为：决策树其实就利用数据结构中常用的树形结构来构造一个预测模型，既然是树形结构，那么必然有根节点、分支节点和叶子节点。里用决策树进行分类时，首先从根节点开始，沿着分支选择下一个子节点，每选择一个节点就利用该节点的分类规则进行分类验证，直到达到最终的叶子节点，此时决定出的分类决策就是该样本所属的类别。构造决策树时首先选择一个根节点，在决策过程中持续添加新的叶子节点作为分类节点，重复该过程直到训练集中所有的数据都被成功的分类才停止。

ID3 和 C4.5 是经典而常用的决策树算法，同时，针对决策树算法存在的不足

也有很多改进算法，分类回归树算法是典型代表。接下来我们针对这几个算法进行简要的过程介绍。

3.2.1 ID3 算法

ID3 算法[45]的理论基础是香农的信息论，其构建决策树时叶子节点的分类标准是样本的信息熵与信息增益。信息熵准衡量了某个随机系统 *S* 所含有的平均信息量，计算公式如式 3-3：

$$Entropy(S) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i) \quad (3\text{-}3)$$

根据香农的信息论，系统的熵与系统的有序性为反比例关系，即系统越稳定，熵越小，也就是系统包含的信息越少。所以说，信息熵反映的是一个系统的稳定度的大小，信息增益和信息熵类似。假定分类训练样本集合为 *S*，其中的某个属性为 D，其信息增益$Gain(S, D)$定义如式 3-4：

$$Gain(S, D) = Entropy(S) - \sum_{v \in Values(D)} (|S_v|/|S|) Entropy(S_v) \quad (3\text{-}4)$$

其中$Values(D)$是 *S* 中属性 *D* 的取值范围集合，S_v表示根据 *v* 在 *S* 中抽取出的全部的 *D*。

概括来讲，该算法的主要思想是尽可能多的计算样本集数据的特征信息熵增，将结果排序后用含有最高信息增益的属性初始建立决策树时的第一个节点，也就是根节点，此时的根节点包含了样本信息的最大信息量，故而可以将样本数据按照最大化的程度进行分开来构建决策树，对于根节点后的子节点同样选择最大信息增益的属性，并将此子节点作为子树的根节点递归重复以上过程构建决策树，直到所有子集的类别相同为止。由于以信息熵为算法的基础，所以 ID3 极大的缩减了最终决策模型的平均深度，从而加快了决策过程，这在对大规模数据进行分类时提升效果明显。但是 ID3 也有一定的缺陷，由于算法的度量标准是信息熵增益，那么根节点的选择必定会倾向于选择信息量最多的属性，但这样的属性并不一定是对样本描述最好的属性，同时，还存在选取的属性比较离散，关联性较低等问题。

3.2.2 C4.5 算法

和 ID3 相比，C4.5[42]主要用来对 ID3 无法处理的连续值属性进行分类。因此，C4.5 构建决策树时将最大的信息增益率作为根节点和叶子节点。该算法的的计算过程分为以下几步：

Step1: 假定有训练集 *S*，其某个连续值属性为 *D*，首先利用离散算法将 *D* 进

行离散后得到一个离散值序列，根据值的大小做升序排列，得到离散属性序列 $D'=\{D_1, D_2, D_3, \ldots, D_{n-1}\}$；

Step2：向 D'中利用分割法将序列随机分为 n 个子序列，通过对 D_i 和 D_{i+1} 的取平均，得到的结果就是分割点 i 的值，经过 i 处理后，D'上就形成了许多小的子区间。

Step3：对 i 求解其信息增益和增益率，前者的计算法和 ID3 一样，这里不再赘述，这里主要一下针对信息增益的改进——信息增益率，其值可以通过式得到式 3-5：

$$GainRario(S,D)=\frac{Gain(S,D)}{SplitInfo(S,D)}=\frac{Gain(S,D)}{-\sum_{i=1}^{c}\frac{|S_i|}{|S|}\log_2\frac{|S_i|}{|S|}} \quad (3\text{-}5)$$

$SplitInfo(S,D)$描述了属性 D 对 S 进行分类判断时的覆盖度以及平均程度，在求得 i 的信息增益率后，值最大的分割点作就是 D 的最优分割点。计算所有的属性，选择增益率最大的属性作为决策树的树根构建决策树。

Step4：构建根节点后，将该节点对应的分割点所划分的取值区间用作新的属性离散序列，重复 Step2 和 Step3，来构建子树，当所有的子区间中的样本分类相同时，决策树构建完成。

Step5：当完成训练得到决策树后，将其用作对未知样本分类的分类器使用。

该算法是众多的决策树算法中比较重要和核心的算法，具有决策规则决策准确度高、简单、构建过程进行剪枝处理从而平衡性好等特点。但是，反复对训练集进行离散、分割和扫描必然会增加 C4.5 构建决策树和预测的时间复杂度，如果数据样本过大，计算损耗也会增加。此外，C4.5 算法对于非内存内的数据无法进行分类处理。

3.2.3 分类回归树算法

分类回归树[30]（Classification And Regression Tree，CART）是对传统分类树算法的进一步改进，具体表现在一下几个方面

1：和传统的回归树算法构造多叉树不同，CART 是构造二叉树来进行决策。

2：针对属性值的离散和连续，CART 最终生成的决策树也不同，如果目标属性值离散，那么最终的决策树是分类树，如果目标属性值连续，那么最终生成的决策树是回归树。

3：CART 在一开始构建决策树时忽略最重生成树的深度和节点的个数，只根据节点的错误率来进行分支处理，在构建完成后再对树进行整体剪枝优化，以提高分类树的推广性。由于是采用二叉树，所以最终的决策树结构更加明了，可以

很直观的明白训练模型时对样本的分类标准和判断准则。

总结起来，CART 算法的运行过程就是先利用样本集依据判定标准构建一颗初始二叉树，之后通过交叉和测试集验证来对二叉树做剪枝处理，达到实现对决策树进行优化的目的。基尼（$Gini$）系数是用来在 CART 构建初始决策树时反映每个类别下节点之间的差异化大小。其具体的公式定义如式 3-6 所示：

$$Gini（s） = 1 - \sum_{i=0}^{k-1} P_i^2 \qquad （3\text{-}6）$$

其中：K 是我们选取的样本集类的个数，K_i 是具体的每个类，S_i 是训练集中属于 K_i 的样本的个数，$P_i=S_i/S$ 是 K_i 在训练集中的相对频度，$i=1$，…，$K-1$。$Gini$ 系数体现的是对 S 进行分区的彻底性。例如，假设对类别 A 和类别 B 进行预测的话，$Gini（s）\in [0，0.5]$，那么$Gini(s)=0$代表 S 中所有的数据都只属类别 A 或者类别 B 于，这样的 S 就是被彻底分类的样本集，$Gini(s)=0.5$意味着 S 中的数据值在这两个类上是均匀分布的。把 S 按照某一分割点进行分割后的每个子集又有自己的$Gini$系数，且不同每个子集再次分割后的子集权重不同，将这些$Gini$系数通过加权求和得到的就是每个子集的$Gini$系数，具体的公式定义如式 3-7：

$$Gini_{split}(s) = \sum_{i=0}^{m-1} \frac{n_i}{n} \times Gini（s_i） \qquad （3\text{-}7）$$

其中 m 表示分区后子集的样本数，n 表示总样本量。

算法首先从根节点开始对训练集进行分割操作，训练集中的所有属性求解$Gini_{split}$后选择结果最小的属性用作根节点和下一个分割点，然后对上述步骤循环来进行子树的构建。

3.3 支持向量机算法

支持向量机（Support Vector Machines，SVM 是一种建立在统计学习理论基础上的较新的机器学习算法，具有学习能力强、适用范围广的优点，是机器学习算法的典型代表 [35]。在以往的分类算法实践中，传统的分类算法暴露出了训练集样本不足、维度过高、或者不满足线性均匀分布时造成的分类偏差、模型训练资源耗费过大、过拟合等问题。当样本集的特征值情况发生变化时，SVM 选取的核函数也会随之改变，并用新的核函数进行分类模型训练，很好的解决了上述存在的问题，目前 SVM 已被大量使用于语义检测、图像识别等领域，取得了可喜额成果。

3.3.1 统计学习理论

统计学习理论[32]（Statistical Learning Theory，SLT）顾名思义，就是借助统计学的研究方法来帮助完成机器学习的研究，目的在于针对大量的样本中存在的暂时无法被有效证明的规律进行归纳和建模，并用其对今后类似的数据进行行为预测。该理论力求通过有限的样本数据归纳出合适的可以被泛化和推广的统计模型以使模型在根据相同的数据进行预测时可以得到最优解。SLT 的核心理论是求解一个函数集，其所能打散的最大样本个数被称为该函数集的 VC 维[32]，反应的是该函数集的适用性。因此函数的 VC 维越高，其适用范围和覆盖面也就越广。

SLT 在确定函数集后，需要对函数集的经验风险与实际风险进行验证分析，在处理二分的问题上，式 3-8 是经验风险 $R_{emp}（w）$ 与实际风险 $R（w）$ 的关系描述：

$$R(w) \leq R_{emp}(w) + \sqrt{\left[\frac{h\left(\ln\left(\frac{2n}{h}\right)+1\right)-\ln（\delta/4）}{n}\right]} \qquad （3\text{-}8）$$

其中 h 是函数集的 VC 维，n 是样本数。

从上式可以发现，二分问题中，实际风险主要是由经验风险和置信区间构成的，并且 VC 维的高低以及训练集的数据规模直接影响到前者的大小。其具体的关系如式 3-9：

$$R(w) \leq R_{emp}(w) + \phi（h/n） \qquad （3\text{-}9）$$

公式（3-9）表明样本数量有限时，SLT 的置信区间、函数集的 VC 维这二者具有正比例关系，当函数集的 VC 维变高，SLT 的置信区间以及复杂程度就会增大，实际风险增加也就会随之变高，此时，通过 SLT 得到的分类模型就会存在过拟合的问题。SLT 指出在分类模型构建时，要同时尽可能的降低 VC 维和减小置信区间并最小化训练集的经验风险，从而让分类模型具有较低的实际风险和较好的分类与推广性，。

3.3.2 SVM 算法

1.广义最优分类面

SVM 的核心思想是利用一个最优的分界将一个可以线性分割的样本集或者非线性的高维样本空间进行类别划分。如果上述描述过于抽象，本文通过对一个简单的二维平面上分布的节点进行分类的图例来将其思想具象化。如图 3-1 所示，平面 S 中分布着大量的黑点和白点，这些点表示了样本集中两种不同的数据样本

类别，直线 L 将这两类样本分开，L_1、L_2 分别表示经过各自类别样本的直线中和 L 垂直距离最小的点并且和 L 不相交的两条直线，L_1、L_2 间的垂直距离表示的是将两种类型的样本节点分离的最小间距。直线 L_1、L_2 经过的点就是将样本分开的边界节点，也就是支持向量（Support Vector， SV）。在求得 L_1、L_2 的时候要保证在完全分离两类样本数据的条件下还要具备最大的垂直距离也就是分类间隔，保证其最大也就保证了模型在面对未知样本集做分类预测时的准确度和适用性达到最大。

我们假定 L 的求解方程为$x\omega + b = 0$，对其作归一化处理后，方程对线性可分的样本集（xi，yi），i = 1， …，n，$x \in R^d$， $y \in \{+1, -1\}$满足式 3-10：

$$y_i[(w \cdot x_i) + b] - 1 \geq 0, \quad i = 1, \ \ldots, \ n \qquad (3\text{-}10)$$

此时平面的分类间隔等于$\frac{2}{\|w\|}$，当 W^2 取得最小值的时候分类间隔$\frac{2}{w}$将取得最大的结果，此时将两种节点按照最小间距分开的界面就是广义最优分类页面。

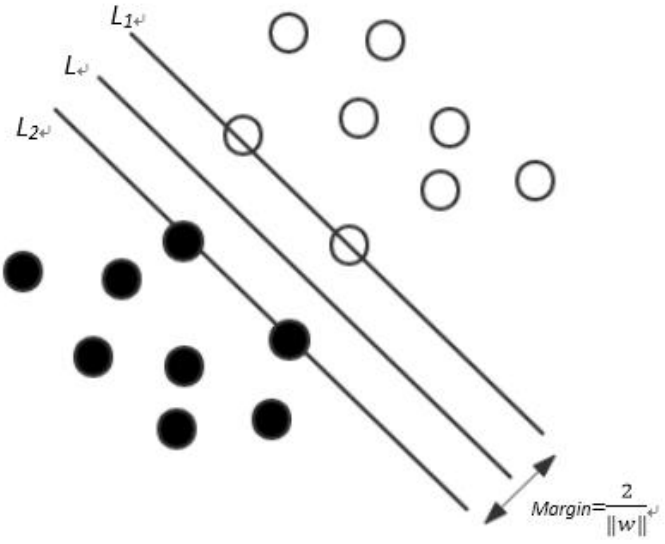

图 3-1 通过广义最优分页面将两种节点分开的示例

2.支持向量机

如果样本集的维度较高，那么其对应的线性函数 VC 维就是 N+1，当 N 取值较大时，此时，为了提高所求分类页面的适用性和准确度需要对将其从较高的维度降低到较低的维度。此外，还可以通过对偶求解来替换之前的求解过程，从而使得原本求解最优分类页面的复杂过程被样本的支持向量机个数简化，从而达到高维空间模型进行广义最优分页面求解的目的。

如果样本集无法线性划分，那么可以先将样本集进行升维处理，在较高的维度进行线性划分后再通过降维操作利用核函数简化高维空间的向量运算来求解最优分类面内积。已知 Mercer 定理如下：

定理 3.1：Mercer 定理核函数 K：$K（u，v）=\varphi（u）\cdot\varphi（v）$

低维空间的核函数集就是所有符合该定理的将地位空间数据映射到高维空间的函数的集合。

当且仅当对于任意满足公式为有限值的函数$\int g（x）^2 dx$为有限值的函数$g(x)$，则：

$$\int K（x，y）g（x）g（y）dxdy \geq 0 \qquad （3\text{-}11）$$

因此，从而简化低维非线性样本空间最优分类面的求解过程，可以通过内积核函数$K（x_i，x_j）$将低维非线性分类问题变成高维线性分类问题。式 3-12 是转换后的目标函数：

$$Q(\alpha)=\sum_{i=1}^{n}\alpha-\frac{1}{2}\sum_{i,\ j=1}^{n}\alpha_i\alpha_j y_i y_j K（x_i，x_j）\qquad （3\text{-}12）$$

相应的分类函数变为：

$$f(x)=sgn（\sum_{i=1}^{n}\alpha_i^* y_i K\left(x_i，x_j\right)+b^*）\qquad （3\text{-}13）$$

综上所述，即使是非线性样本集，SVM 也可以通过维度提升来使其线性可分，然后在通过降维简化来求得非线性样本集中数据的最优分类面。

3.SVM 算法中的核函数

SVM 算法中用到的和函数常见形式有以下几种：

（1）多项式核函数：$K\left(x，x'\right)=(\alpha x\cdot x'+\beta)^q，\ \beta>0$ （3-14）

（2）径向基核函数（RBF）：$K\left(x，x'\right)=\exp\left(-\gamma\left||x-x'|\right|^2\right)，\ \gamma>0$ （3-15）

（3）Sigmoid 核函数：$K\left(x，x'\right)=tanh（\gamma x\cdot x'+c）$。 （3-16）

这三种核函数中，第二种因为有原理简单、分类正确性高、适用性广等优点而得到更广泛的使用。

3.3.3 SVM 算法的主要优点

相比于传统的分类算法，SVM 具有以下几个显著的优点：

（1）由于 SVM 是建立在 STL 原理的基础上，因此可以通过利用结构化风险最小化来讲经验风险降到最低。

（2）既可以求解最小分类距离，也可以求解最大分类距离，这有效的避免了出现模型过度学习问题的可能。

（3）与只关注局部最优解的一些分类算法不同的是，SVM 求出的最终结果满足局部最优性的同时也满足全局最优求解的要求。

（4）可以通过内积核函数将分线性可分的低维样本集进行高维映射转换为线性可分的数据集，在通过降维求解最优分类面，从而在解决非线性可分空间的分类问题的同时还能保持较低的算法复杂度。

3.4 本章小结

本章选取了朴素贝叶斯算法、ID3 算法、C4.5 算法、分类回归树（CART）算法以及支持向量机（SVM）算法这五种经典的分类学习算法，详细分析了这些算法的基本定义、核心思想、计算过程以及各自的优缺点。

4 恶意链接检测系统的设计和实现

本章内容在基于前三章介绍的基础上，来具体介绍和实现恶意链接检测系统通过制定系统的设计原则和实现目标来规划系统的设计模块，并对这些模块做详细的实现方法介绍。

4.1 系统的设计目标和原则

系统的设计就是把需求转变为功能，并通过设计规划好系统的架构和工作流程，实现所有的功能和非功能性需求。本论文研究的主要内容是设计一个恶意链接检测系统，该系统即可以将原始的恶意链接数据进行数据去重后，通过特征匹配，SVM 分类过滤出恶意链接，也可以通过实时监测用户浏览页面的 DOM 结构变化，来检测恶意 JavaScript 脚本做出的恶意攻击，并将这些检测结果存储更新黑白名单库。系统在设计时主要遵循以下几条设计原则：

1、开放性：系统的整个代码设计遵循开源代码协议 MIT。

2、标准性：系统要采用开放式体系架构，各项技术遵循行业和相关规范。子系统之间高内聚、低稠合，降低相互间依赖度，保证系统正常使用。

3、扩展性：扩展性系统设计要考虑到未来发展的需要，尽可能设计得简明，降低各功能模块稠合度，并充分考虑兼容性。系统能够支持对多种格式数据的存储

4、成熟性：系统要采用业内主流、成熟的体系架构来构建，实现跨平台的应用。

5、高效性：要规范代码编写，精简逻辑，并选择较为底层和高效的语言来实现系统。

系统的设计目标如下：

1、实现动静结合的检测功能，即既可以直接针对 URL 的特征来判断其是否为恶意链接，也可以通过实时监测页面 DOM 结构的改变来检测恶意 JavaScript 脚本中存在的恶意链接。

2、既要支持传统的基于特征和规则的过滤，也要借助分类算法和机器学习技术来建立自适应分类学习模型，克服传统检测引擎对攻击的变种应对不够及时的缺点。

4.2 系统模块结构和工作流程

1、系统的模块

根据对恶意链接和恶意站点的特征进行提取分析，为了使系统对已知类型的恶意链接保持高度的准确性降低误报率的同时还能够针对变种后的恶意链接攻击做出及时的自动学习和识别，同时还可以针对隐藏在恶意 JavaScript 脚本中通过延时加载逃避检测的恶意链接进行筛选检测，本文设计和实现了基于 SVM 和 DOM 检测算法的恶意链接检测系统(SD-MLDS)，SD-MLDS 系统利用基于黑/白名单库将已知类型的恶意链接进行检测和剔除后，再通过使用机器分类学习技术对未知的新形式的恶意链接利用通过经验样本集提取特征进行训练学习得到的分类器进行恶意链接检测。SD-MLD 系统共分为三个子模块，分别为:基于黑/白名单的恶意链接检测模块，基于 SVM 的恶意链接检测模块和基于 TDD 算法进行恶意暗链接提取的检测模块。SD-MLD 系统结构如图 4-1 所示：

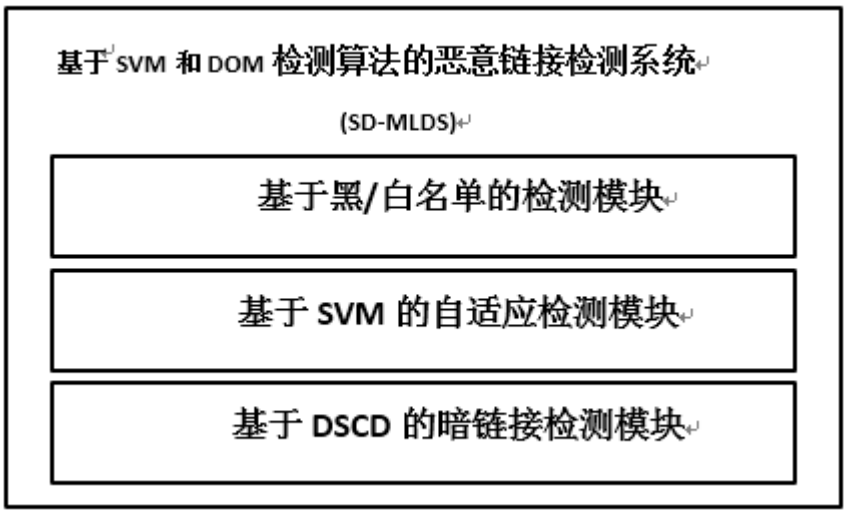

图 4-1 基于 SVM 和 DOM 检测算法的恶意链接检测系统

以下是几个模块具体功能的介绍：

（1）基于黑白名单库的特征匹配检测模块：该模块首先需要根据每天定期获取的恶意链接建立和更新一个黑/白名单库，之后利用字符串匹配技术、哈希值比对技术等将命中黑/白名单库中样本的链接进行匹配检测。

（2）基于 SVM 的检测模块：由于黑/白名单库中的链接都是已知的恶意链接和非恶意链接，因此可以将其作为经验样本集，提取样本集中数据的属性特征向量，通过对特征向量的学习来构建和更新恶意链接分类预测模型作为分类器来对测试样本集中的链接进行恶意检测，并将结果更新黑/白名单库，利用新的黑/白库来对分类模型进行增量自适应更新。

（3）基于 TDD 算法的检测模块：通过应用 TDD 算法实时监控页面 DOM 结构的改变，将新增的 DOM 节点识别出来，提取其中 JavaScript 脚本中存在的暗链接，并将该链接利用前两个模块来判断是否是恶意链接。

2、系统的整体运行流程

本系统的工作流程图如图 4-2 所示：

Step1：系统读取格式化后的恶意链接样本集，经过数据排重过滤掉重复的数据。

Step2：将去重后的数据，首先进过基于黑白名单库的模块进行规则检测。

Step3：通过检测的链接集将流入基于 SVM 的恶意链接检测模块进行特征分析和学习检测，一旦发现新的恶意链接，模块将该链接放入黑名单库，并将更新的黑名单作为新的样本集训练、更新恶意链接分类模型。

Step4：系统会将前两步中既不属于黑白名单也不能完全确定不是恶意链接的 URL 放入内嵌的 WebView 引擎模块进行解析并实时监控 DOM 的变化状态，一旦发现 DOM 结构改变则将新增的结构抽取，分析其中的脚本部分是否包含 URL，并将其输入到检测模块进行恶意链接复检。

此外，更新后的黑白名单库将作为接口提供给反病毒引擎等程序使用。

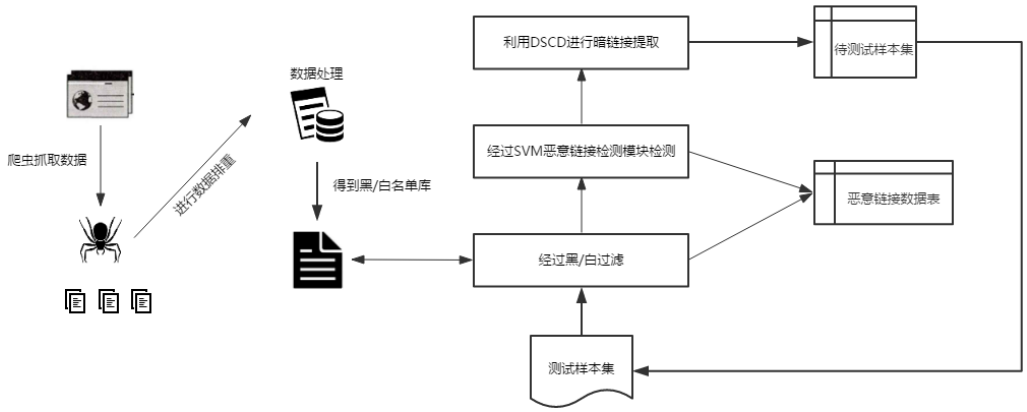

图 4-2 基于 SVM 和 DOM 检测算法的恶意链接检测系统的工作流程图

4.3 开发环境

本文中关于系统的设计与实现部分，采用的工具为：

开发环境： VS2012

数据库：Postgres 和 Redis

发布程序：Tomcat

4.4 数据排重模块设计

从本系统设计的角度来看，数据排重[37]的目标是对重复和相似的恶意链接进行去除，精简样本集中的数据，通过减少数据冗余来降低后续流程计算的数据量。这里的难点有两个：（1）数据量大，需快速完成计算，性能要求高；（2）描述同一个事件的文本内容不完全相等，需要柔性比较。从目前技术的普及程度来看，很容易想到使用基于哈希的 MD5 排重技术。

从效率角度老考虑，如果利用 MD5 进行恶意链接排重，其需要先将恶意链接数据集中的所有数据一遍通过哈希函数计算后存储在一个 MD5 链中，一边进行 MD5 码的比对，那么，每一次的比对存储都需要从 MD5 链的第一个位置开始依次比较，时间复杂度为 $O(n)$，这显然不满足大规模数据清洗的要求。因此，可以通过建立一个 MD5 值的索引数据表来简化比较过程，达到 $O(1)$ 的时间复杂度。但是，索引的引入需要一定的存储空间，且该索引必须在查重时维护在内存中。那么，当恶意链接数据样本集增大，该索引表也必将随之增啦，消耗大量宝贵的运行内存，极端情况下，如果数据表的大小超出内存范围，那么超出部分必须用磁盘等外部设备存储，这必定会增大磁盘的读取 IO，极大的影响系统的效率。

从识别的柔性方面，我们很容易发现，由于是基于内容做计算，那么任何两句话，只要有标点符号不一样都不会排重。因此，其存在着缺乏柔性，排重率低的问题。

针对这些问题，其改进技术，标准布隆过滤器被提出和应用。

4.4.1 布隆过滤器

布隆过滤器[8]（Bloom Filter）是一种经典的数据信息查找匹配技术，以速度快、效率高、准确性好而在文件查找、P2P 节点搜索得到了广泛的使用。其核心思想是通过一组彼此之间独立不相关联的哈希函数，将数据集中的数据映射到一个多位向量中去，因此，布隆过滤器查重所需的存储空间与待查数据样本的数量以及单个数据自身的大小无关，而与选取的位向量的位数有关。假设利用一个 m 位向量来对含有 n 个元素的数据集合进行存储，那么平均每个元素需要的位数是 m/n，假设每一位的大小是 h，那么存储这 n 个数据所需要的空间为 hn。因此，利用布隆过滤器进行数据查找，可以有效的降低数据存储的成本。

布隆过滤器查找的具体过程如下：

Step1：设有含有 n 个元素的数据集合 $x=\{x_1, x_2, \ldots, x_n\}$，含有 K 个相互独

立的哈希函数的函数集和 $h=\{h_1, h_2, \ldots, h_n\}$，以及一个由 m 位初始值为 0 组成的位向量 V。h 和 m 共同组成了一个布隆过滤器，如图如图 4-3 所示。

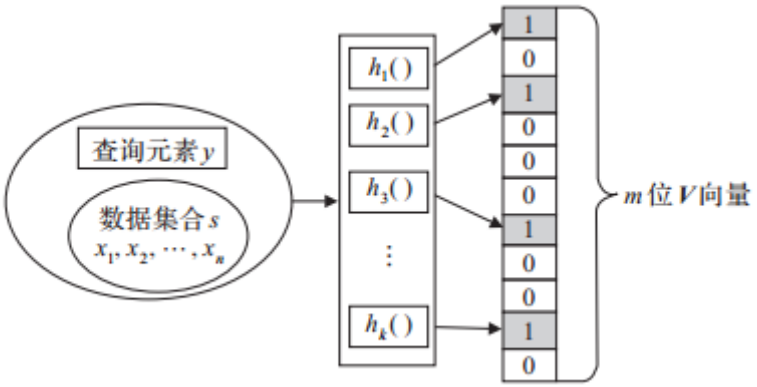

图 4-3 布隆过滤器结构示意图

Step2：从集合 x 中依次去除数据，对每一个数据，利用 h 进行位向量映射后转换成为一组由 0 和 1 组成的数据集合，将该集合中所有的 1 存储到对应的位向量 m 中。存储时，如果 m 中某一个位置的值已经为 1，那么就忽略这一位的存储，否则就将其置为 1。如图 4-4 中，x_1，x_2 都映射到了同一个向量位，但该位置只被 x_1 首次置 1。

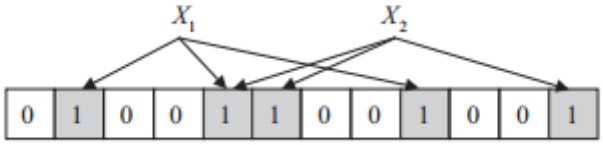

图 4-4 布隆过滤器插入元素

Step3：设有一待查数据集 $y=\{y_1, y_2, \ldots, y_n\}$，检测 y 中是否有数据属于集合 x 时，和存储类似，利用哈希函数集合 h 对 y 中的每一个数据进行映射，并将映射后的集合中所有值为 1 的数据在 m 中进行查找，如果 m 中所有对应位置的值都为 0，那么说明否则说明该数据不属于集合 x，反之，则可能属于集合 x。如图 4-5 所示，y_1 不是集合 x 中的元素，y_2 可能属于该集合。

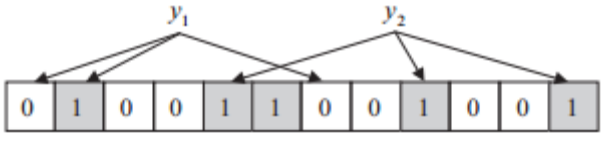

图 4-5 查询元素

以上就是布隆过滤器的存储和查找原理。虽然布隆过滤器存储简单，查找快速，但是存在假阳性（False Positive）误判的缺点。证明如下：

哈希函数按照等概率随机选择将向量的某一位（假设为 m_j）置为 1，，则该位没有被置为 1 的概率是就是$1-\frac{1}{m}$，经过哈希函数集合 h 中有的函数计算后都

没有被置为 1 的概率为$\left(1-\frac{1}{m}\right)^{k}$。对于含有 n 个数据的集合 x 来说，所有数据都经过映射后，该位依然没有被置 1 的概率为$\left(1-\frac{1}{m}\right)^{kn}$，则被置 1 的概率为$1-\left(1-\frac{1}{m}\right)^{kn}$。

当 n 个数据都被映射存储后，当利用这个位向量进行查重时，被查的数据 y_i 经过 K 个哈希函数计算后都将 m_j 置为 1 的概率为$\left(1-\left[1-\frac{1}{m}\right]^{kn}\right)^{k}\approx\left(1-e^{-kn/m}\right)^{k}$。

以上证明是建立在这 k 个哈希函数彼此独立的前提下的。由证明可以看出，与位向量的位数 m 成正比，与数据集合的数量 n 成反比。因此，当 m 和 n 确定时，哈希函数集合的大小 k 可以通过$\frac{m}{n}\ln 2\approx 0.7\frac{m}{n}$求得。

此时假阳性的概率 p 为：$2^{-k}\approx 0.6185^{m/n}$

而当 p 的大小确定是，位向量的最优位数 m 可以通过$m=-\frac{n\ln p}{(\ln 2)^2}$求得。

以上分析表明，由于$\ln p<0$，因此，m 和n成正比比，即位数组增大，输入的数据规模也可以随之增大。如果 m，n，k 为定值，那么此时p的最大值为：$\left(1-e^{-k(n+0.5)/(m-1)}\right)^{k}$。

布隆过滤器牺牲了很小的准确率，换来的是存储成本的极大降低和查找速度的较大提高。因此，在对误判率要求不高，而对存储和查找速度要求很高时，可以选择布隆过滤器作为折中的查重算法。

4.4.2 MSHASH 算法

从上一小节对布隆过滤其的分析可以发现，如果利用布隆过滤器对恶意链接进行存储和查重的话，需要对查找的数据进行全文指纹判断，无法实现柔性判断，而且其位向量位数的大小和以及哈希函数集合的大小会对误判率有较大的影响。针对以上问题，本文首先考虑利用柔性比较算法 SimHash 算法[9]进行改进。SimHash 算法的核心思想通过降维来将高维的特征向量映射成一个低维的特征向量，通过比较特征向量之间的汉明距离[9]（Hamming Distance）的大小来来确定数据是否相似或者相同。其具体的步骤如下：

Step1：对给定的数据进行特征提取得到有效的特征向量。

Step2：对每个特征向量计算哈希值，并将得到的 n 位 01 值作为数据的签名。

Step3：因为不同的特征在描述数据时的权重不同，因此，对第 2 步得到的计算结果再按照特征的权重计算得到加权签名。

Step4：将所有的加权签名累加的到数据的文本签名。

Step5：如果文本签名中某一位的数值大于 0，则将该位置为 1，反正则置为 0，得到该数据的最终特征向量。

Step6：计算两个数据最终特征向量的汉明距离，根据结果来判断其相似程度。

图 4-6 是该算法的详细图解。

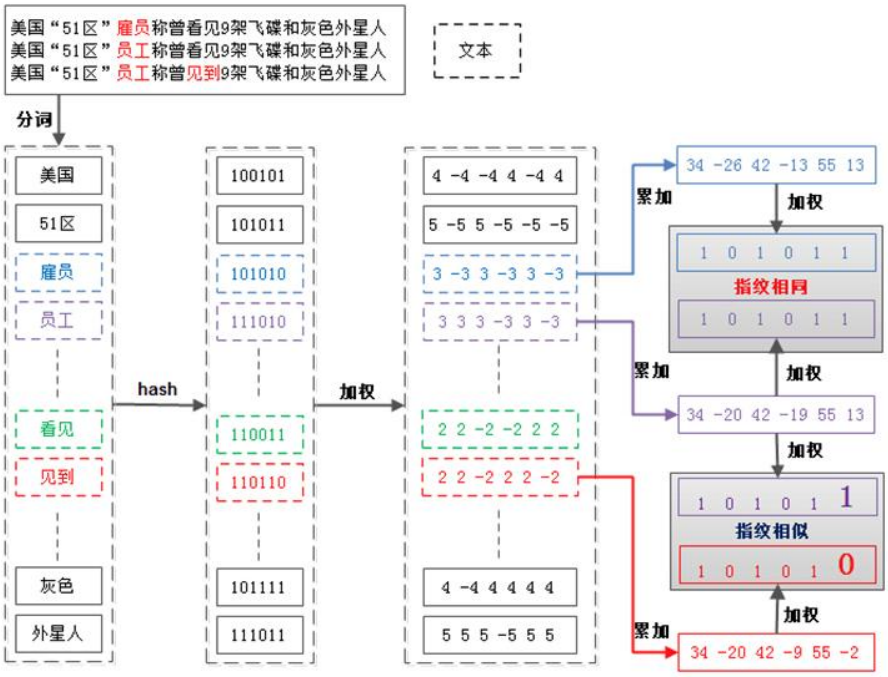

图 4-6 SimHash 算法解析图

SimHash 算法实现了柔性比较，但是计算汉明距离的耗时在整个算法中占比较大，对算法的速度起决定性影响。根据 Sadowski 等人[9]实验验证，如果用单机对 300 万的数据进行两辆比较的话，需要 4 个小时的处理时间，这样的速度显然无法满足实际的业务要求。因此，本文借鉴局部敏感哈希算法(Locality-Sensitive Hashing，LSH)的思想，对 SimHash 进行了一点改进，提出了 MSHASH 算法。MSHASH 的基本思想是先利用分桶将肯定不相似的数据分到不同的桶中，在每个筒内的数据再利用 SimHash 进行量量比较，求解其之间相似度。其主要步骤下：

Step1：利用随机算法将 SimHash 计算出的签名值随机排列。

Step2：利用可以返回签名中首次出现 1 的位置的自定义函数进行位置求解，也就是 MSHASH 值。

Step3：将 MSHASH 值相同的数据分在一个桶内，进行两两比较。

图 4-7 是该算法的详细图解

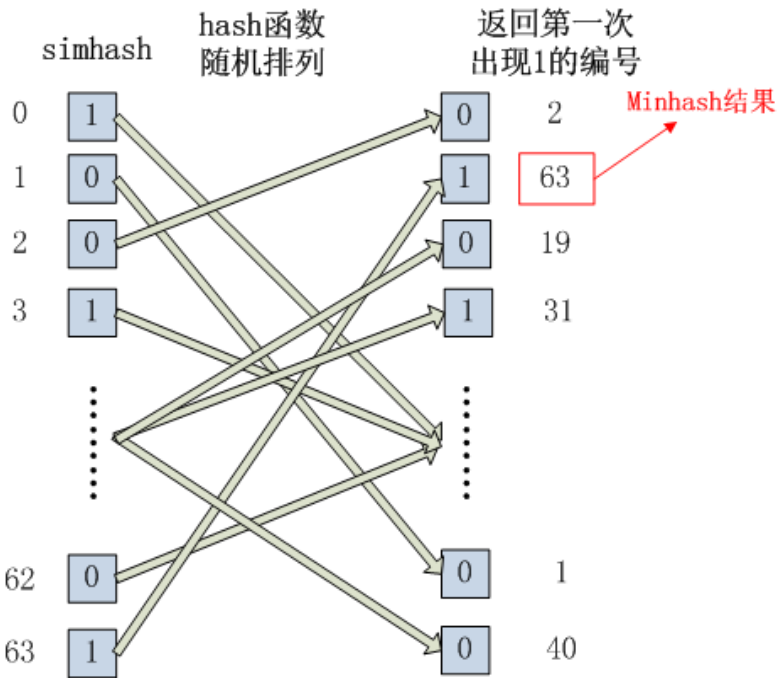

图 4-7 MSHASH 算法解析图

MSHASH 算法思想中：

第 1 步，如果两个数据通过 SimHash 得到的签名中出现 1 的个数越多，那么两个数据经过自定义函数求得的 MSHASH 值相等的概率也就越大。

第 2 步，通过用多个自定义函数进行多次随机排列计算 MSHASH，可以控制相似的程度。

4.4.3 MSHASH-BF 排重技术

综合上述分析，我们结合布隆过滤和 MSHASH 二者的特性，最终选择 MSHASH-BF 排重技术。首先通过 MAHASH，通过较大的分桶粒度快速筛选出相似数据集，之后利用布隆过滤对该集合做细粒度的数据排重。该法既拥有 MSHASH 的柔性特性，又具备了布隆过滤高效节约存储耗时短的优点。

4.5 恶意链接检测模块设计

4.5.1 基于黑/白名单库的恶意链接检测模块

该模块的核心是通过包含已知恶意链接 URL 地址和特征的黑/白名单库来对一个链接是否是恶意的进行判断。该模块对待检测的未知链接进行检测，若未知链在黑名单库中匹配成功，那么该链接就必定是恶意链接，同时，将该链接记录

到系统的关系型数据库当中去。

该模块使用的黑/白名单库中的黑名单存储的是已经被确定的恶意链接，白名单中的链接则是经过安全认证的正常站点。下面将具体描述这两个子模块的设计与实现。

具体的检测过程是，将待检测的未知链接样本集输入到该模块的入口后，利用上一小节提出的 MSHASH-BF 算法，将样本集中的数据先与白名单中的 URL 进行匹配，如果匹配成功这此链接是安全的站点地址，针对该链接的匹配结束，否则，将其与黑名单中的 URL 匹配，如果成功匹配，那么该链接就是恶意链接，需要拦截并记录到系统数据库中的结果检测表中，针对该链接的检测完成。以上步骤成功匹配的链接无论是恶意的还是安全的都要从样本集中删除。如果经过上述匹配后，样本集中依旧存在链接数据，那么这些链接就是未知类型链接，此时的样本集就是位置类型链接样本集，将其流转到后续的检测模块进行检测处理。整体的检测流程图如图 4-8 所示：

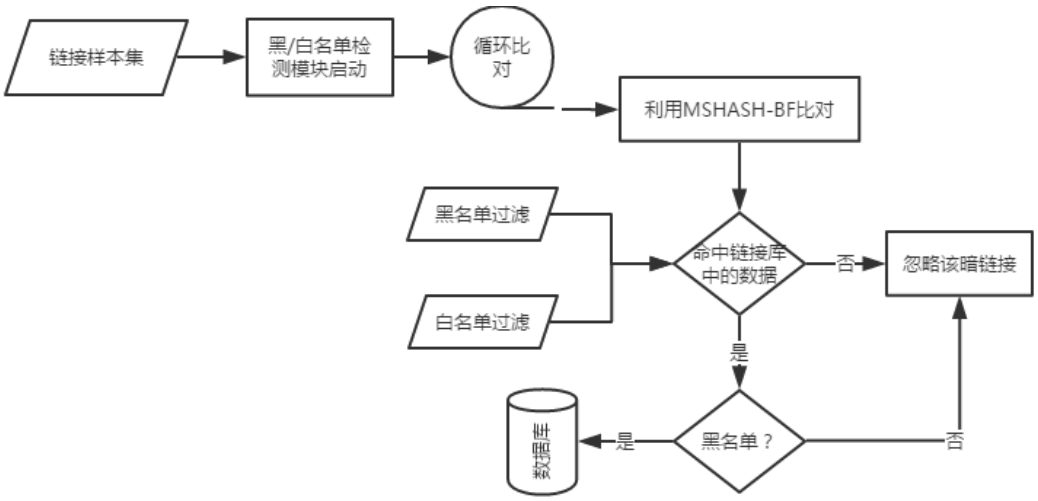

图 4-8 于黑/白名单库的恶意链接检测模块工作流程图

具体的检测步骤如下:

Step1：输入待测数据样本集

Step2：利用 MSHASH-BF 技术先将样本集与白名单匹配，匹配成功的从样本集中删除。

Step3：利用 MSHASH-BF 技术将白名单过滤后的样本与黑名单库进行匹配，匹配成功的样本将被记录到系统的关系数据库中，并同样要被从样本集中去除。

Step4：如果进过第二步和第三步的处理过后，样本集为空，那么本次匹配过程结束，否则，将该样本集作为未知类型链接的样本集，流转到后续的模块来检测处理。

4.5.2 基于 SVM 的恶意链接检测模块

互联网的飞速发展，大量新技术的涌现给黑客提供了许多新的攻击途径，同时，多年的网络安全对抗不仅丰富了网络安全工作人员的经验，也让黑客在应对新的安全检测系统时具有更快的攻击演变能力。因此，每时每刻，网络上都会出现新的恶意链接形式。在这种情况下，仅仅采用类似基于知识库和特征匹配等的被动防御技术已将无法满足实际的安全需要。因此，我们需要利用自动分类技术，通过将经验样本集内的数据进行特征提取构造分类预测模型，来对未知类型的链接进行安全检测。由于链接的判定具有二分性，即要么是恶意要么是非恶意，因此，本系统采用 SVM 算法来构建系统的分类引擎。

SVM 的核心是通过在线性可分的训练集上求解一个最优分类面来讲数据进行分类，并且通过低维空间到高维线性空间映射后降维来处理非线性可分的样本分类问题。算法具体的原理本文在第三章有详细的介绍，这里不再赘述。实际应用场景下，我们的样本集数据量是有限的，SVM 通过有限的样本集训练后的分类模型具有模型简洁、速度快、准确率高等特点，且基于统计理论又使得其具有很强的适用性。但是，需要注意的是，为了做到自适应，系统还需要将通过 SVM 预测出的新恶意链接与的旧的恶意链接训练集进行整合更新，并用更新后的训练集重新训练分类模型。

4.5.2.1 模块流程

基于 SVM 的恶意链接检测模块的整体工作流程如图 4-9 所示：

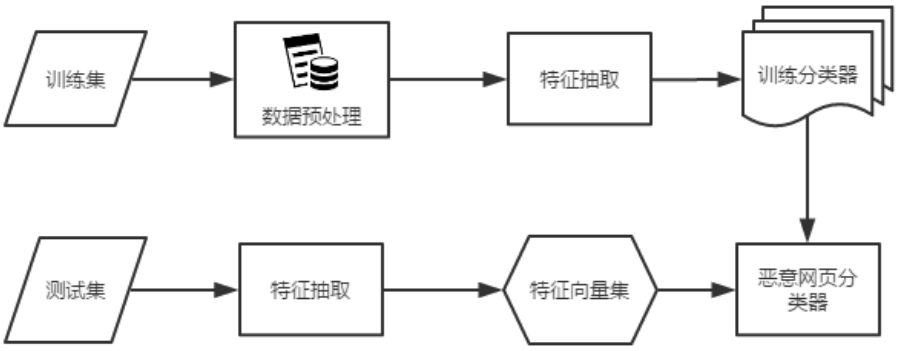

图 4-9 基于 SVM 的恶意链接检测模块的整体工作流程图

从上图我们可以看出，本模块的具体检测步骤分为以下几步：

Step1：将训练集数据经过数据清洗、数据变化等方式做预处理。数据清洗的

目的主要是讲训练集中各种不同编码的链接统一起来，并且还可以消除冗余信息和噪声。数据集成则是将多个不同的训练集进行合并和统一。

通过这一步可以在保持原训练集特征的前提下最大限度精简数据，从而加快模型的训练，简化生成的分类模型，这对于本模块最终的实际预测效果至关重要。

Step2：将清洗后的数据通过特征抽取引擎对链接从特殊字符、链接长度等角度进行特征抽取。

Step3：利用抽取的特征和样本集利用 SVM 进行分类模型训练，生成的模型作为模块的分类器。

Step4：利用分类器对新输入的未知链接进行分类预测，并将结果记录到系统数据库，同时将检测出的恶意链接作为新的样本集，用来和更新经验样本集。

4.5.2.2 URL 异常特征分析和选择

目前，网络上的恶意链接数量难以估计，常见的黑名单库中的恶意链接数据就有数百万之巨，如何提取有效的特征成为了一个技术难点。基于 SVM 的恶意链接检测算法，链接的特征选择的正确与否，对最终的生成的分类器和分类结果的准确性有直接的影响，因此，提取出合适的恶意链接特征成为了本模块的一个关键点所在。由于恶意链接的攻击目的包括但不限于钓鱼、种植木马、扫描信息等并且隐藏的方式多种多样，因此，恶意链接往往具备多种不同的离散的特征。

关于特征向量的提取国内外许多学者做了很多实验，Garera 等人从结构角度分析提取了恶意链接的 18 个结构特征；黄华军等人在考虑结构特征的前提下又引入了 4 个词汇特征。为了尽量保证恶意链接特征提取的准确和完整性同时精简生成的分类模型，本文在通过分析常见的恶意链接形式并结合前人多年的研究基础上，选取了恶意链接的 6 个结构特征、7 个账户信息特征、7 个商标名特征、11 个短链接特征、5 个移动网络下的特征共 36 个特征组成一个特征向量作为系统的预设特征，用来作为生产分类器的特征标量，具体特征如下：

（1）F_1：由于用户习惯于通过域名访问站点，大部分用户缺少对 IP 概念的理解，因此，许多恶意链接会通过 IP 地址的形式隐藏自己的域名来诱导用户访问。

（2）F_2：链接中含有多个“$”、“@”等特殊的符号；

通过利用普通用户对基本网络安全知识的缺失，黑客往往会利用一些特殊符号来构造恶意链接对用户进行欺骗和攻击，例如 https://wvw.baidu.com@211.12.167.334。如果用户点击了该链接，那么浏览器在解析该 URL 的时候将会忽略，实际上合法的网址 URL 很少含有@等特殊字符，还

包括“-”、“#”、“￥”等。

（3）F_3：链接中包含 3 个及以上的“.”；

通过二级及以上的域名对用户进行隐藏欺骗是恶意链接的常用手段之一，实验验证，人们在浏览网页时，对域名中字符的关注个数不会超过 12 个，同时，如果用户是在移动设备上浏览网页，由于受限于移动设备的屏幕大小，这个数字还会更低。因此，许多恶意链接通过添加多个“.”来构造二级域名模仿其它网站来诱骗用户访问。

（4）F_4：链接的总字符数超过 22 个；

根据 Freebuf 网站 2015 年的发布的《恶意网址常见特征分析报告》，恶意链接的平均期望长度一般为 22 个及以上，而正常的站点出于易访问性、SEO 性能等角度的考虑，其链接长度的均值不会超过 15 个字符。

（5）F_5：URL 文件路径有"http"字符；

HTTP 协议是 Hyper Text Transfer Protocol（超文本传输协议）的简称，互联网上网站的服务器通过此协议向用户的浏览器的传送超文本，是互联网的基础性协议，也就是我们常看到的链接中的协议头标识。由于传统的 HTTP 协议是明文传输，出于安全角度考虑，HTTP 协议经过 TLS 或者 SSL 协议进行加密处理后就成了 HTTPS 协议。因此，链接中出现 HTTP 或者 HTTPS 就意味着浏览器需要将请求发送到该协议之后定位的位置，是一种重定向标识。黑客往往利用特点，将自己的恶意地址通过 HTTP 和 HTTPS 协议与正常的链接进行伪装，例如 https://www.baidu.conm/search/http://www.attacks.com 就是一个典型的通过重定向进行攻击的恶意链接。

（6）F_6：含有大量 Base64 编码

Base64 是一种用来传输 8 位字节码的编码方式。由于 HTTP 协议中 GET 请求对请求的参数等有一定的限制，因此，Base64 常常可以用来传输较长的数据信息，经常运用于前端图片的压缩等方面。然而，黑客也会利用这种技术，结合重定向、二级域名欺骗等将真正的恶意链经过 Base64 编码隐藏，来欺骗用户。例如 https://wvw.baidu.com @211.12.167.334 经过编码伪装后的形式为 https://wvw.baidu.comQDIxMS4xMi4xNjcuMzM0，具有很强的欺骗性。

（7）F_7-F_{13}： 链接中包含 account，webscr、pay、login、sign in、banking、confirm。

以上的 7 个特征是 Garera 等人通过实验验证抓取到的恶意链接常常包含的字符串，可以发现，这些子符基本都是和用户的账户信息有关，这直观的反映出了黑客进行恶意攻击想要达到的目的。

（8）F_{14}-F_{20}：链接中包含 qq、paypal、alipay、weixin、taobao、tmall、163；

由于黑客极其容易将知名的社交网站和电商网站作为自己的攻击目标。根据《2016 年 6 月钓鱼网站处理简报》，金融支付、在线购物、社交网络、电子邮箱这几类是钓鱼链接模仿的主要目标，这几类恶意链接占了整个钓鱼链接的 98.79%，国内像新浪微博、淘宝、天猫等知名网站占了攻击总体的 68.2%。因此，本文选择以上 7 个商标特征。

（9）F_{21}-F_{31}：链接中包含 sina.lt、 t.cn、 dwz.cn、 qq.cn.hn、 tb.cn.hn、 jd.cn.hn、 tinyurl.com、goo.gl、 is.gd、j.mp、bit.ly；

近几年来，随着移动互联网的发展和社交网络的兴起，短平快的即时分享需求越来越被各种内容提供商看重，受限于微博等社交网络发表内容字数的限制，同时也为了在移动设备上节省更多的屏幕空间、让用户更方便的分享内容，短链接的技术得到了发展。简单来说，短链接就是通过计算将长 URL 转换为简短的地址。然而，这一技术也被黑客所利用，通过短链接来隐藏自己的恶意网址。例如，https://www.attacks.com 经过短链接的转换后就成为了 http://t.cn/RaM7ecj，用户完全识别不出这是恶意链接。但是这样的链接却可以在新浪微博传播。经过实验分析，目前常见的短链接中主要包含以上 11 个字符串，因此，选取这些字符串作为特征进行提取。

（8）F_{32}-F_{36}：链接中包含.apk、ipa、mobile、webapp、scheme；

近些年，移动设备得到了长足的发展，普及度广，性能强，成为了现代人浏览互联网的主要端设备。丰富的应用占据了人们的碎片化时间。这样的背景下，通过恶意链接进行恶意软件下载成为了黑客的一个攻击目标。由于移动设备屏幕大小有限，难以显示全部链接地址，因此，通过长链接、编码等方式将恶意软件的下载地址进行伪装欺骗用户点击下载成为了移动互联网下的一种攻击方式。此外，研究表明，用户在每个 app 上停留的时间平均不超过 6 分钟。这意味着，用户在使用移动设备时经常进行应用切换操作。黑客利用这一用户习惯，通过利用非 HTTP 的伪协议来构造恶意链接让用户点击，从而呼起手机内其它的恶意软件等进行攻击。例如，scheme://www.attacks.com?sigT=6a4f8sdad&sigU=attack.ipa 就是一个典型的第三方伪协议，目的在于呼起用户手机内安装的 attack 应用。

基于以上原因，本文选取这 5 个作为移动网络下的恶意链接特征。

本文在选取这 36 个特征后对其进行赋值表示，由前文所述，链接的分类具有二分性，即非恶意和恶意，因此，我们采用 0、1 来代表两种结果，即“1”代表链接包含该特征，可能是恶意链接，“0”则相反。然后，我们利用函数表达式来抽象表示这些特征向量，具体的表示如式所示：

$$F_1 = \begin{cases} 1 & if\ \ URL\ \ is\ \ IP\ \ address \\ 0 & others \end{cases} \qquad (4\text{-}1)$$

$$F_2=\begin{cases}1 & if\quad UTL\ contains\ \text{"@","\$" etc.}\\ 0 & others\end{cases} \tag{4-2}$$

$$F_3=\begin{cases}1 & if\quad URL\ .>3\\ 0 & others\end{cases} \tag{4-3}$$

$$F_4=\begin{cases}1 & if\quad length(URL)>22\\ 0 & others\end{cases} \tag{4-4}$$

$$F_5=\begin{cases}1 & if\quad URL\ \text{contains "http" or "https"}\\ 0 & others\end{cases} \tag{4-5}$$

$$F_6=\begin{cases}1 & if\quad Base64(URL)\\ 0 & others\end{cases} \tag{4-6}$$

对于盗取用户账户信息的恶意链接特征，我们用F_i进行表示

$$F_i=\begin{cases}1 & if\quad URL\ \text{contains "login" etc}\\ 0 & others\end{cases}$$

$$i\in\{7，8，9，10，11，12，13\} \tag{4-7}$$

对于模仿知名网站钓鱼的恶意链接特征，我们用F_j进行表示：

$$F_j=\begin{cases}1 & if\quad URL\ \text{contains "alipay" etc.}\\ 0 & others\end{cases}$$

$$j\in\{14，15，16，17，18，19，20\} \tag{4-8}$$

对于利用社交网络传播的恶意链接特征，我们用F_k进行表示：

$$F_k=\begin{cases}1 & if\quad URL\ \text{contains " t.cn" etc.}\\ 0 & others\end{cases}$$

$$k\in\{21，22，23，24，25，26，27，28，29，30\} \tag{4-9}$$

对于移动页面恶意链接特征，我们用F_l进行表示；

$$F_l=\begin{cases}1 & if\quad URL\ \text{contains " apk" etc.}\\ 0 & others\end{cases}\qquad l\in\{24，25，26\} \tag{4-10}$$

4.5.2.3 自适应 SVM

网络中每一天都会出现许多新的恶意链接形式，其数量也逐日增加，因此，仅仅根据经验样本集数据训练得到的分类模型依旧无法满足日新月异的恶意链接攻击技术。为了解决这一问题，本系统将预测出的新的恶意链接反馈进黑名单库，并将更新后的黑名单库作为新的样本集训练进行属性特征提取和学习，从而更新分类器。因此，解决该问题的本质如何实现经验训练集和测试训练集样本的

同步。

目前，自适应分类学习主要的实现思路有以下几种：

（1）利用自适应滑动窗口在样本中随机选择一段数据，并给该数据的出现时间和特征重要进行赋权，将所有选择出来的数据按照加权计算后按照从大到小排列，选择最大的样本与经验样本进行合并更新后在重新训练。

（2）直接将测试集和经验集进行合并更新后重新训练。

（3）将不同样本集训练出来的分类器按照时间、准确性、适用性等进行加权合并生成新的分类器。

以上的思路都有一个共同的不足之处，就是需要重新用样本集训练，这在数据量很大时会造成系统的资源消耗过多。Yang 等人[31]借鉴规范损失最小化理论，提出了自适应支持向量机算法(Adaptive Support Vector Machine，ASVM)。ASVM 无需更新原有的经验样本集，可以用新增样本集直接增量训练更新分类模型。也就是说，AVSM 摆脱了经验样本集的限制，因此，相较于之前的思路，ASVM 具有极强的适用性和实际应用价值，及其适合需要通过增量更新来维持服务运行的业务场景。

ASVM 的原理为：假设$S^p = S_l^p + S_u^p$代表含有部分标记过得的原始数据，S_l^p标识的数据已经被正负标签标识，也就是训练集数据，与之相对，S_u^p就是没有进行标识过得未知数据，也就是测试集数据。$S_l^p = \{(x_i, y_i)\}_{i=1}^N$，其中$x_i$代表第 i 个样本的向量，$y_i$是该样本的提取的特征标识，$y_i \in \{-1, +1\}$。$S^a = \{(x_i{}^a, y_i{}^a)\}_{i=1}^{N^a}$代表新增的样本集，$y_i{}^a$代表新样本集中$x_i$的标签。$f(x)$表示利用训练集进行特征属性分析后得到的分类预测模型。假定通过 ASVM 学习新增训练集后得到新的分类模型增量$f^a(x)$。其具体的计算公式为式：

$$f^a(x) = y_i + \Delta f(x) = f(x) + w^T\varphi(x)$$

w 是新训练集中样本的权值，$\varphi(x)$用来将低维度向量高纬度向量进行转换。w 通过以下式得到：

$$min\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i$$

其中$\xi_i \geq 0$，$y_i f(x_i) + y_i w^T\varphi(x) \geq 1 - \xi_i, \sum_{i=1}^N \xi_i$的作用是表示$\Delta f(x)$。对测试集进行预测所得结果的总误差。通过上式得到的最优平面可以和经验样本集与新增样本集都保持最小的边界距离。C 在这里起到权值调节的作用。将上式按照用拉格朗日改写为式 4-11：

$$L_p = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^N \xi_i - \sum_{i=1}^N \xi_i u_i - \sum_{i=1}^N a_i\big(y_i f(x_i) + y_i w^T\varphi(x_i) - (1 -$$

$\xi_i)\big)$ （4-11）

其中, a_i和u_i都是大于零的常数，在对ξ和 w 求导后就能得到L_p的最小解，即式 4-12：

$w = \sum_{i=1}^{N} a_i\, y_i \varphi(x_i)$，$a_i = C - u_i, (i = 1,2, \dots, N)$ （4-12）

从上式可以知道，$\Delta f(x) = \sum_{i=1}^{N} a_i y_i\, k(x_i)$用作再生希尔伯特空间(RKHS)的核函数，该空间的内积公式为式 4-13：

$$\left\| f^a - f \right\|^2 = \|\Delta f\|^2 = \langle \Delta f | \Delta f \rangle = \sum_{i=1}^{N} \sum_{j=1}^{N} \xi_i a_i y_i\, a_j y_j K\big(x_i, x_j\big) = \|w\|^2 \quad \text{（4-13）}$$

从上式可以发现，$\left\| f^a - f \right\|^2$与$\|w\|^2$相等，因此 $f^a(x)$和 $f(x)$之间的距离达到最小值。

同时，式（4-11）要是想取得最小值的话，还必须满足一下几个条件：

$$a_i\{y_i f(x_i) + y_i w^T x_i - (1 - \xi_i)\} = 0$$

$$a_i \geq 0$$

$$y_i f(x_i) + y_i w^T x_i - (1 - \xi_i) \geq 0$$

$$u_i \xi_i = 0$$

$$\xi_i \geq 0$$

$$u_i \geq 0$$

将公式(4-4)带入公式(4-3)中就得到了拉格朗日二元目标函数：

$$L_D = \sum_{i=1}^{N} (1 - \lambda_i) a_i - \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{N} \sum_{j=1}^{N} \xi_i a_i y_i\, a_j y_j K\big(x_i, x_j\big) \quad \text{（4-14）}$$

其中，$\lambda_i = y_i f^a(x_i)$，$a = \{a_i\}_{i=1}^{N}$可以在a_i满足$C \geq a_i \geq 0$将L_D做最大值求解来处理，即求解L_p的最小值。

仅仅通过增加一个λ，ASVM 就提升了通过新增训练得到的分类模型的预测效率和精度，而且，λ只有在新增的训练集第一次进行属性特征学习时求解，之后的所有运算过程，该值都将保持不变。因此，通过 ASVM 用新增训练集增量训练得到的$f^a(x)$较之前的全量更新$f(x)$在算法的时间和物理开销上减少很多。

4.5.3 基于 DSCD 算法的恶意链接检测模块

前两个模块的运作需要有一个前提，就是恶意链接地址已经被抽取获得，即检测的是显式链接。但是，许多链接通过利用懒加载、延迟加载、暗链接的形式隐藏在网页中，之前的检测思路在解决这种问题时就显得捉襟见肘。因此，我们设计了基于 DOM 结构改变检测算法（DOM Structure Changing Detection， DSCD）的恶意检测模块来实时监听页面 DOM 结构的变化，针对新增的 DOM 片段进行抽取，利用正则表达式将其中的暗链接检测出来后，输入到前两个模块进行恶意链接检测。

系统的整体流程图如图所示：

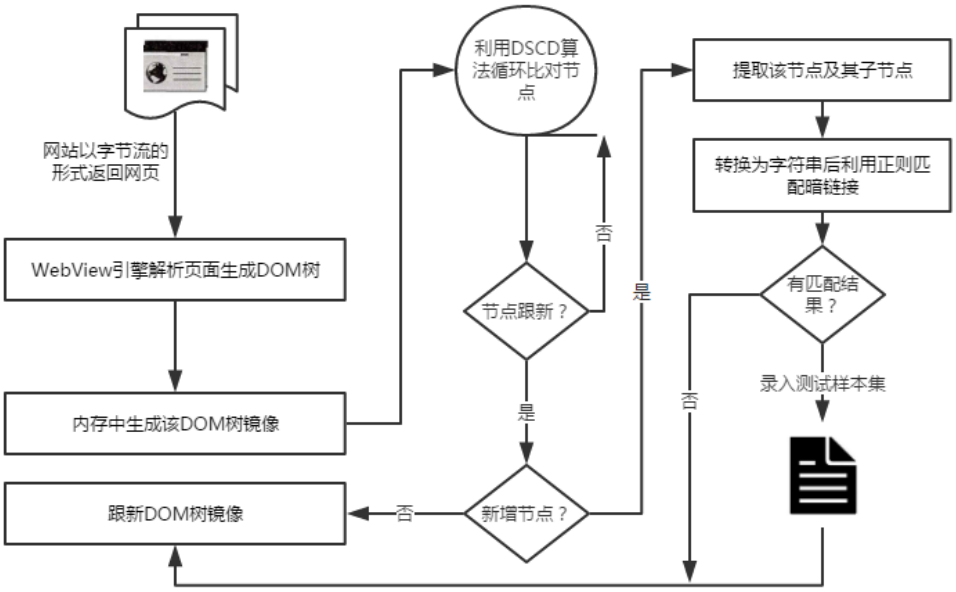

图 4-10 DSCD 模块流程图

其中，该模块又分成两个子模块，暗链接检测和 DOM 结构监听

4.5.3.1 暗链接检测

所谓暗链接[37]，就是指没有在网页上直接按照 URL 地址的形式表现出来的链接，通常会通过弹窗广告、JavaScript 脚本加载和重定向等方式隐藏，在用户不在意的情况下将用户导向钓鱼等恶意页面。

针对这一情况，本文根据暗链接常见的分类，构建不同的正则表达式，将隐藏在网页代码中的暗链接进行提取，并分离出暗链接中的域名信息，将其和当前页面的域名进行匹配，如果发现两者的域名不相同，就将其放入系统维护的待测试样本集中，具体的流程图如 4-11 所示：

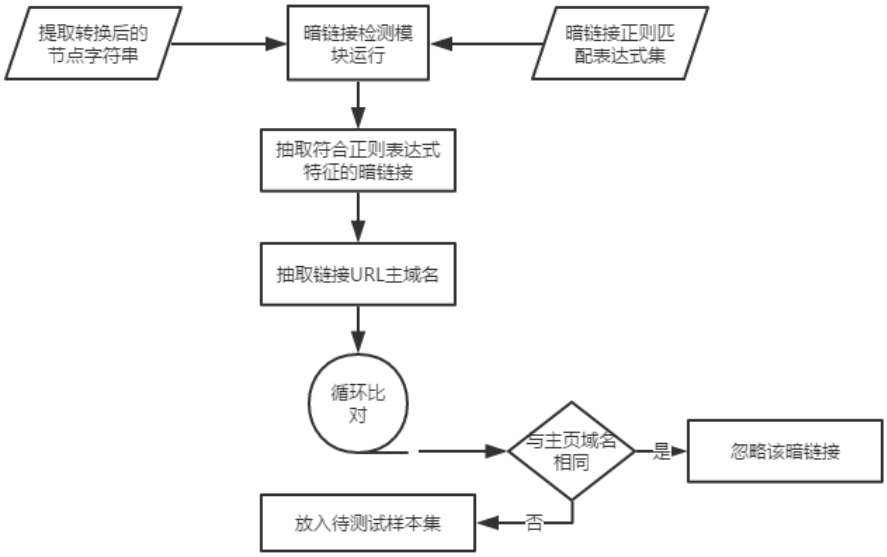

图 4-11 暗链接检测模块流程图

由上图可知暗链检测子模块的工作流程为:

经过对常见暗链接的分析并结合前人的研究成果，本文将暗链接按照常见的特征和对应的正则表达式分成以下几种：

（1）利用 iframe 标签隐藏:

通过 AJAX 请求返回 JavaScript 脚本插入 iframe 后加载恶意脚本代码或者运营商利用 DNS 劫持给页面插入 iframe 后加载广告页等是常见的暗链接加载形式，此时的 iframe 会通过将 width/height 设置为 0，或者将 display 属性设置为 none、visibility 设置为 hidden 等方式在页面上隐藏，虽然 iframe 不被浏览器渲染引擎渲染，但其通过 src 引入的外部脚本依旧会被解析引擎加载和解析执行。针对这种形式的暗链接，我们构建如下正则表达式：

```
/[^>!]*((width\s*?=\s*?0)|(height\s*?=\s*?0)|(visibility:\s*?hidden)|(display;\s*?none)|(style\s*?=\s*?"hidden")|(top\s*?:\s*?-[0-9]{3,}\s*?px;)|(left\s*?:\s*?-[0-9]{3,}\s*?px;))[^>]*/ig
```

（2）超长字符串:

经过前文分析可以知道，恶意链接和正常的网页链接相比在长度上有很大的不同，常见的恶意链接往往具有很长的地址用以迷惑用户和躲避系统检测。因此，我们将长度大于 50 的链接作为潜在的恶意链接，并构建如下正则表达式：

```
/[0-9A-Za-z]{50,}/ig
```

（3）异常字符和特定字符串:

和链接长度异常类似，包含异常字符和特定的字符串的链接也是标识潜在恶意链接的重要标签。根据对常见的恶意链接进行分析，总结出如下异常字符：@、$、^、~、<、>、|等，以及如下特定字符串：downloader、shellexecute、http、

shellcode 等。因此，我们构建如下正则表达式：

```
/ [\@\$\^~\<\>]|(downloader)|(http)|(shellcode)|(http)|(shellexecute)/ig
```

（4）Document.write:

根据 ECMA 标准，浏览器的 DOM 对象具有 document.write API 用来支持动态的添加 HTML 结构。只要将 HTML 文本转换成字符串对象输入该函数，浏览器就会解析这些字符串，渲染其描述的 DOM 结构。黑客通常会将暗链接借助 iframe 隐藏后将其转换成 HTML 文本字符串的形式诱使用户浏览器加载。例如：

```
<script>(document.write("
    <iframe src='http://www.attacks.index/attack_index'>")
</script>
```

其中 http://www.attacks.index/attack_index 返回的是一个恶意页面，该页面中会内嵌恶意脚本来通过 iframe 等标签动态加载恶意网页的内容。由于该 API 频发使用会对浏览器的开销较大，因此该函数很少在正常网页的开发中使用，但是经常会被黑客利用来进行攻击，因此可以将该函数出现的次数大于 5 次作为其可能存在暗链接的重要特征。针对此情况，我们构建如下正则表达式：

```
/[document\.write\(\s\)]{50}/ig
```

4.5.3.2 DOM 结构监测

由于暗链接需要通过 iframe 等利用懒加载技术注入页面，这必然会导致页面 DOM 结构的改变，因此我们只要随时监控页面，通过 DSCD 算法找到新插入的 DOM 节点后提取，并利用上文中的暗链接正则表达式进行暗链接提取，得到提取结果后就可以作为待测试样本集留待系统检测。由于 DSCD 是该处理过程的关键，因此，本文对该算法做详细的阐述。

DSCD 算法基于著名前端 MVVM 框架 VUE 中的虚拟 DOM 比对算法[24]进行改进，该算法的核心思想是内存中维护的页面 DOM 快照与实际的 DOM 页面进行同层节点比对，寻找新增 DOM 结构的根节点。具体过程为：

我们假定有两颗 DOM 树 M、N，分别代表内存中的 DOM 快照和新的实时页面 DOM 结构，两颗树中节点我们用 a、b、c、d 进行表示，如图 4-12：

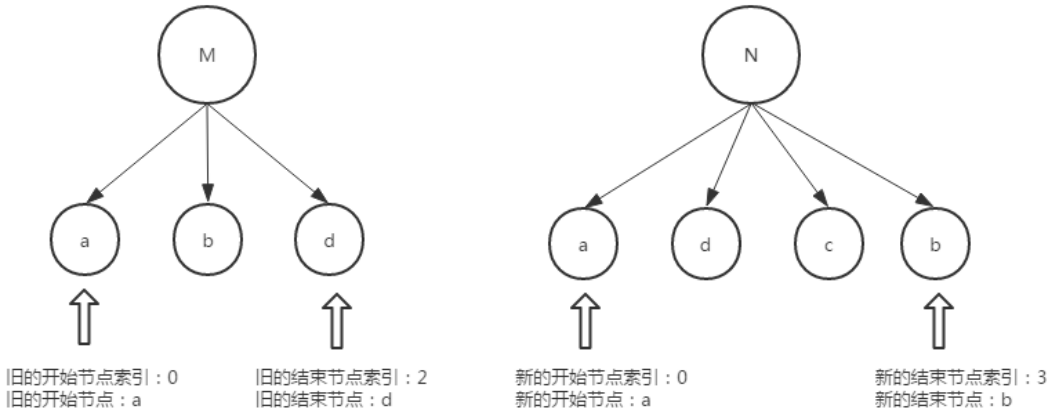

图 4-12 初始状态图

当 N 结构发生改变后，我们开始进行节点比较。此时，新、旧的开始节点为 a，由于节点相同，我们把新、旧开始节点的索引加一。此时的状态如图 4-13：

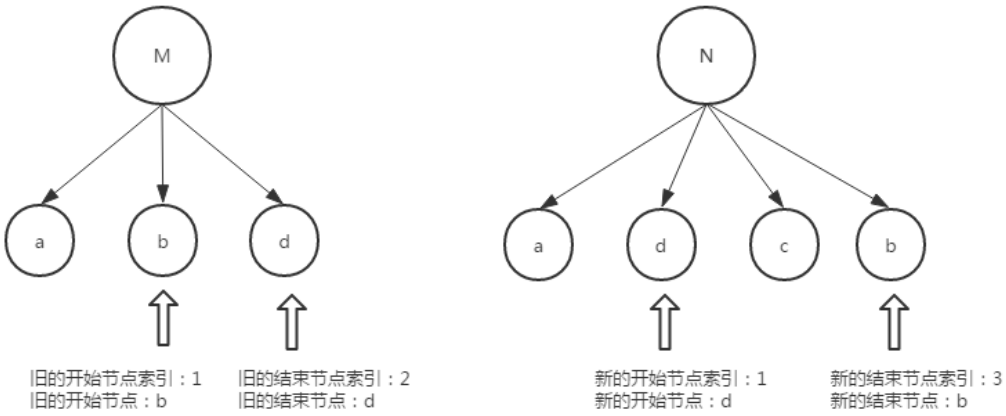

图 4-13 第一次比较后的状态图

比较完 a 节点，我们继续对同层的 b 节点进行比对，此时旧的开始节点和新的结束节点相同，我们将 M 树中的 b 节点更新为 N 树中 b 节点的位置，并将旧的开始节点索引加一，新的结束节点索引减一。此时的状态如图 4-14：

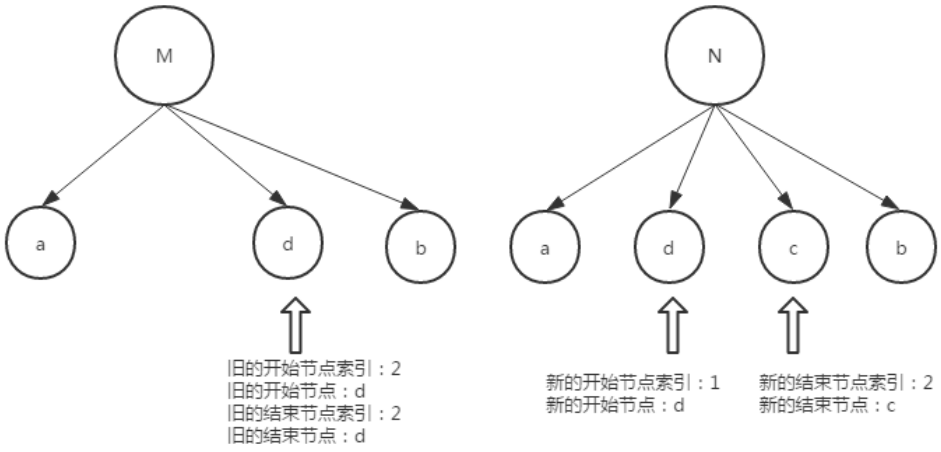

图 4-14 第二次比较后的状态图

进行第三次比较，此时旧的结束节点和新的开始节点相同，都是 d，因此，

我们将 M 树中的 d 节点更新到 N 树中 d 节点的位置，并将就的节点索引减一，新的开始节点索引加一。此时的状态如图 4-15：

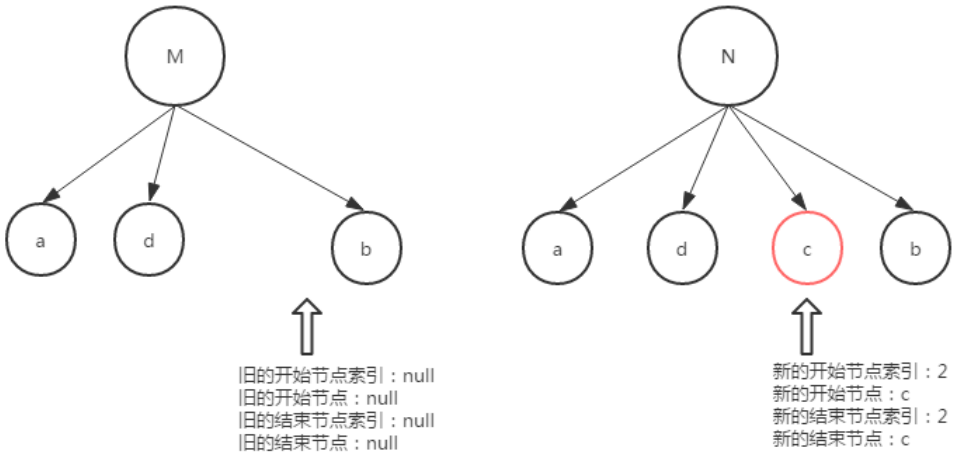

图 4-15 新增节点示意图

进行第四次比较，此时旧的开始和结束节点索引指向空节点，而新的开始和结束节点指向节点 c，因此，c 节点即为新增的节点的根节点，将其下有的子节点提取出来，就是新插入的 DOM 结构。这是因为如果广告是一个叶子结点，则提取出该叶子结点，如果广告是一棵子树，则这棵子树的所有子节点都是广告，提取该子树的根结点。

根据以上两条结论，就可以从文档的根节点开始，对两棵树做同层结点对比。同层节点对比采用广度优先遍历原则，如果该层无新增节点，则采用深度优先遍历原则，选取子节点，然后重复上述步骤即可。

4.6 本章小结

本章详细介绍了恶意链接检测系统的架构设计原则以及工作流程，并详细介绍了数据排重模块、恶意链接检测模块下用到的数据排重算法和分类检测算法。

在数据排重模块，本章详细介绍了常用的布隆过滤器，并针对其存在的不足提出了改进方法，MSHASH 算法和 MSHASH-BF 算法。

在恶意链接检测模块，本章详细介绍了基于黑/白名单库、基于 SVM 以及基于 DSCD 算法这三个检测子模块的工作流程和实现原理。同时，本章分析了标准 SVM 算法在实际应用中存在的缺陷，并针对这一缺陷，引入了 ASVM 算法对分类模型进行增量训练更新来满足系统的设计需求。

5 实验验证和结论

5.1 数据采集

本系统的运行前提需要有一个预设的黑/白名单库。网络安全工程经过多年的发展，已经积累了相当多的对抗经验和数据，许多安全机构都提供了可供第三方使用的黑名单库和恶意链接集。本文的实验数据来源如下

（1）利用 HtmlUnit 随机抓取 www.phishtank.com、绿盟、Freebuf、恶意网站实验室等安全站点上 6012 条恶意链接数据作为测试样本集中的负面数据，

（2）随机爬取 www.websecurityguard.com 上 10674 条数据，平均分成两部分即 5337 条，分别用作测试数据和白名单苦数据，条正常网页作为测试样本集的正面数据。

（3）为了保证黑名单组成数据种类的多样性，保证覆盖率，本文从表 5-1 中的站点随机爬取 20144 条数据经过去重后组成黑名单库。

表 5-1 **恶意链接抓取站点**

编号	网站 URL
1	http://www.malwareMacklist.com
2	http://www.freebuf.com/
3	https://zeustracker.abuse.ch/
4	http://mirrorl.malwaredomains.com
5	https://www.kaspersky.com.cn/
6	http://www.mwsl.org.cn/
7	http://webscan.360.cn/url
8	http://www.txahz.com/topic-shadu.html
9	http://Ustsxlean-nixxom/pipeniiail/viruswatch/
10	http://www.virscan.org/

5. 2 数据排重

由于获取数据的来源不同，必定存在许多重复的样本，因此，需要对抓取的数据进行去重处理。根据第四章介绍的布隆过滤、MSHASH、MSHSHS-BF 技术，本文分别例用这三种技术对组成黑名单库的 20144 条数据进行去重处理，并从时间和准确率两个维度来进行性能对比。

1、布隆过滤器参数选择

哈希函数数量的不同会对布隆过滤器的效果有较大影响，因此，我们需要选择合适的哈希函数个数。对 20144 条数据求其对数，为 14.298，因此，我们选择 14 作为初始值，在[12，18]的期间内进行筛选对比，其结果如图 5-1 和 5-2：

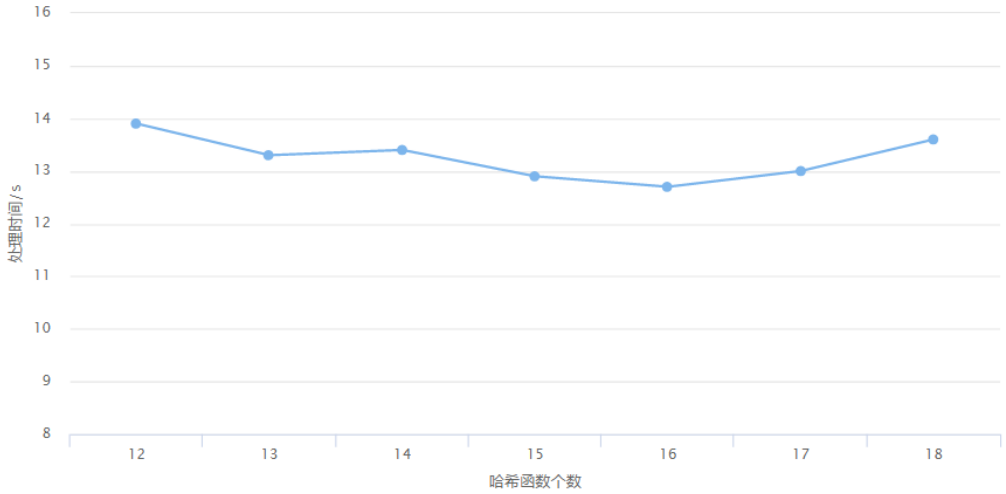

图 5-1 不同哈希函数个数的处理时间

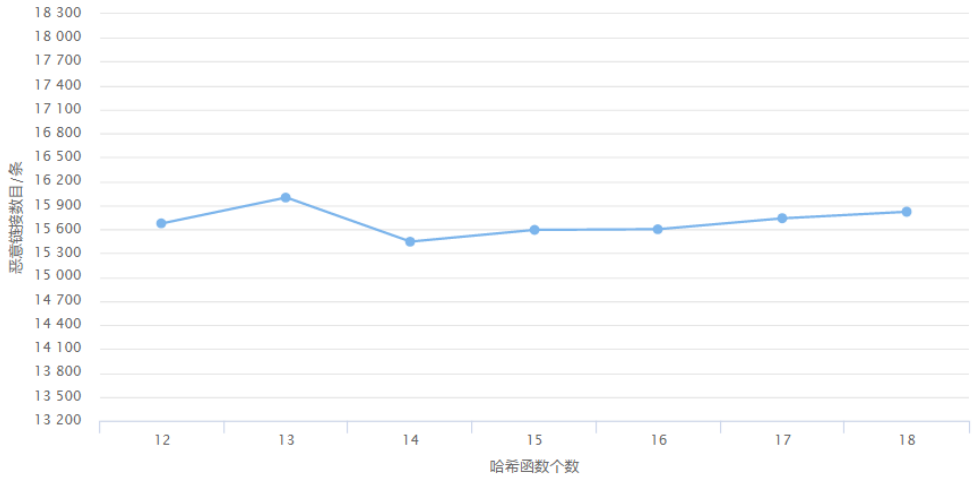

图 5-2 不同哈希函数个数过滤后的数据个数

从上两张图可以看出，选择 15 个哈希函数可以在相对较短的时间内去去的较好的过滤结果，因此，本文将哈希函数的个数设置为 15 个

2、计算有效数据期望

将原数据经过拷贝扩充到 100 万，选择 8 位分桶，利用 map reduce 集群用三种算法过滤，其时间和过滤结果如表 5-2 所示

表 5-2 三种过滤结果对比

算法名称	处理时间/s	处理后的数据数量/条
布隆过滤	107.9s	15932

MSHASH	65.4s	15640
MSHASH-BF	70.3s	15683

可以看到，MSHASH-BF 相对耗时较少的同时可以去的很好的去重效果。

我们比较了 MD5、MSHASH、MSHASH-BF 三种方案的优缺点如表 5-3 所示。

表 5-3 三种排重方法特性比较

方案	优点	缺点
布隆过滤器	速度快	硬性比较
MSHASH	柔性比较	两两比较耗时多
MSHASH-BF	柔性比较，相对耗时较少	结果依赖分桶粒度

5.3 分类结果评价标准

如前文所述，恶意链接检测的结果只有两种：恶意链接和非恶意链接，因此，可以将其看作是一个二值分类判断过程，利用二值分类的评价体系来评估本系统的预测效果。通过将恶意链接假定为 0 或者是负样本，正常链接假定为 1 或者是正样本后，可以得到表 5-4 的恶意链接检测系统检测结果评价表。

表 5-4 恶意链接检测系统检测结果评价表

	实际是恶意网页	实际是正常网页
被判定为恶意网页	*TP*	*FP*
被判定为正常网页	*FN*	*TN*

上表中各符号的含义为：

TP（True Positive）：代表的是被成功检测出来的恶意链接样本数量；

TN（True Negative）：代表被分类到白名单中去的正常链接的样本数量；

FP（False Positive）：代表被识别为恶意链接的正常连接的样本数量；

FN（False Negative）：代表被分类到白名单中的恶意链接的样本数量。

基于以上的分析，可以得到以下几个最终的评价标准：

（1）*TPR*（True Positive Rate）：$TPR = \frac{TP}{TP+FN}$，即成功检测出恶意链接的概率，反应的是恶意链接分类模型的准确度。

（2）*FPR*（False Positive Rate）：$FPR = \frac{FP}{FP+TN}$，即将正常链接分类为恶意链接的样本在所有正常样本中的比例，衡量的是恶意链接分类模型的误报概率。

（3）*R*：$R = \frac{TP}{TP+FN} \times 100\%$，代表实际被检测出的恶意链接的比例，也叫

召回率。

（4）A： $A=\frac{TP+TN}{N}\times 100\%$，代表被成功正确检测的恶意和非恶意链接在所有测试样本集中所占的比例，体现的是分类模型的整体识别准确度。

（5）P： $P=\frac{TP}{TP+FP}\times 100\%$，代表的是所有被判定为恶意链接的结果中真正的恶意链接比例，反应的是分类模型针对恶意链接的准确识别概率。

5.4 测试结果分析

5.4.1 基于黑/白名单的匹配结果

前文可知，本次试验中恶意链接和正常样本分别为 6012 和 5337，正负样本比大约是 1.12:1。经过数据排重后，本文选择 15683 条恶意链接和 5012 条正常链接组成黑/白名单库，并经过滤后的 5419 条测试恶意链接作为待测样本数据。经过匹配过滤，共匹配出 1441 条数据，部分匹配结果如表 5-5 所示：

表 5-5 于黑/白名单的匹配的部分结果

序号	恶意链接	类型
1	http://baenainc.com/data/wpinclude/js/Leo/viewscod/viewscod/	钓鱼
2	http://dentavalo.com/one.php	广告
3	http://www.iperalarm.com/cli/atualizacao/digitau-bankline/index_agente.php	XSS
4	http://www.wachter.cc/webmail/	钓鱼
5	http://www.fortuneo-fr.eu/portail/mabanque/cce426f89734fa973a8ea1b1a3e7ebec/	广告
6	http://redeportais.com.br/eventos/wp-admin/js/gov-impots.client/index/6f4498f86f5bb0c04f8b275bdfa3344c/b0b6aa31a1ee5925a4b3372b2a22ac78/redirection.php	XSS
7	http://vinoinvilla.it/wp-includes/Requests/Auth/en	钓鱼

可以看出，黑/白名单过滤了 26%的恶意链接，远远达不到我们的安全要求。另外，黑白名单需要系统每天爬取大量的数据进行数据去重和合并等维护操作，存在明显的滞后性。

5.4.2 传统分类结果比较

为了实际检测分类算法对测试样本的分类效果，本文将测试样本集按照随机分割均分为十等份，每份约 1000 条数据，每次随机取其中的 9 份集合黑名单库进行分类模型训练，取剩下的一份用来进行测试，将恶意链接标识为 1，正常连接标识为 0，分别对每个实验的实验结果按照 R 和 A 进行对比验证，并取十次结果的平均期望作为最终的预测精度。

根据表 5-4 的分类标准以及 F-Measure，本实验从五个角度来测试和验证朴素贝叶斯、C4.5、分类回归树以及支持向量机这四种经典分类算法针对测试集的分类结果。其分类的最终结果如表 5-6 所示：

表 5-6 四种分类算法结果比较

分类算法	判定	TPR	FPR	P	R	F-Measure
朴素贝叶斯	1	0.612	0.14	0.831	0.62	0.693
	0	0.86	0.388	0.712	0.84	0.79
C4.5	1	0.81	0.13	0.85	0.822	0.833
	0	0.87	0.19	0.848	0.87	0.851
CART	1	0.881	0.09	0.932	0.876	0.909
	0	0.91	0.119	0.881	0.93	0.92
SVM	1	0.898	0.04	0.95	0.90	0.932
	0	0.96	0.102	0.913	0.955	0.939

由上表可以看出：

（1） SVM 的 TPR 是 0.898，使所有的算法中最高的，朴素贝叶斯算法的 TPR 最低，为 0.612，其它两个算法的 TPR 和 SVM 的不大，但是也已经足以说明 SVM 在恶意链接检测方面有较好的表现。

（2）从 FPR 来看，朴素贝叶斯的 FPR 为最高的 0.14，其次是 C4.5 的 0.13，SVM 的 FPR 最低，为 0.04，这表明 SVM 算法的识别误判率比较低，而朴素贝叶斯算法存在较大的误判率。

（3）SVM 算法的 P 值最高，为 0.95，紧随其后的是 CART 的 0.932，最低的依旧是朴素贝叶斯算法，为 0.831。这表明 SVM 在恶意链接判断的准确率方面是四个算法中最好的。

（4）从 R 值来看，SVM 的 0.90 依旧是所有算法中最高的，朴素贝叶斯再次获得了最低值 0.62。

（5）比较几个算法的 F-Measure，SVM 以 0.932 获得第一，朴素贝叶斯为 0.693。

图 5-3 是四种分类算法在测试集上分类后的指标直方图。可以很明显的看出，在五个指标是，SVM 较其它三个算法有明显的优势。

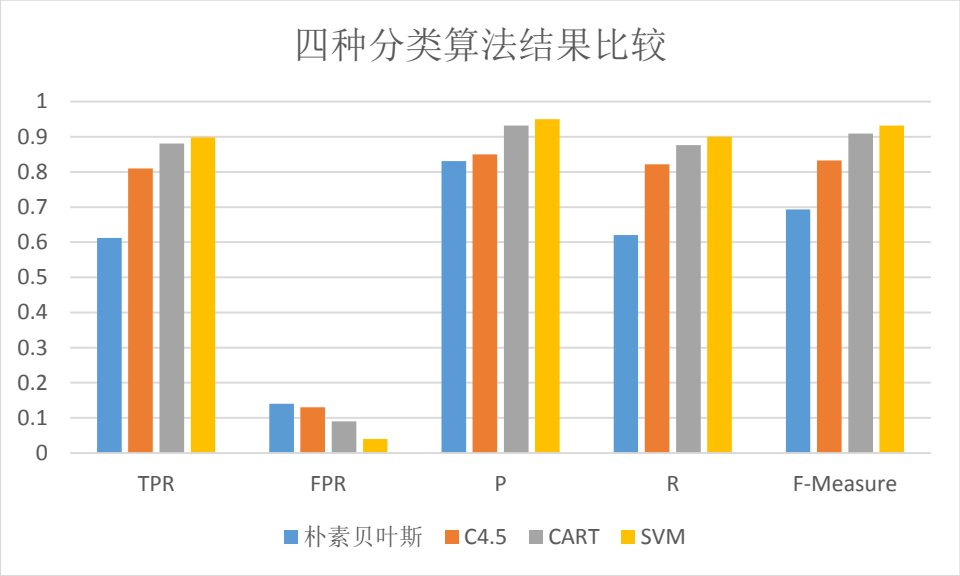

图 5-3 四种分类算法比较结果的直方图

通过以上的对比我们可以得出以下结论：

（1）SVM 在恶意链接识别和检测中可以去的最好的识别效果，并将误判率降到了几种算法中的最低，且较其它算法有明显的领先优势。

（2）C4.5 和 CART 在恶意链接识别中表现居中。

（3）朴素贝叶斯的所有测试指标都是最低的，并且和其它三个算法有较大的差距，明显已经无法满足恶意链接检测的识别要求。分析出现这种现象出现的原因，本文猜测可能是朴素贝叶斯算法是基于样本集属性特征之间彼此独立的前提来进行分类的。

5.4.3 SD-MLDS 结果验证和分析

5.4.3.1 ASVM 分类效果

本小节主要对 ASVM 的分类模型更新后的分类效果进行实验验证。首先，利用爬虫从上文的数据采集网站上上随机抓取 3000 个恶意和正常链接数据，依旧采用交叉验证的思路，将数据按照每组 300 个进行随机划分成十组，分别用 t_i 表示，i 的取值从 1 到 10，随机抽取 9 组作为训练集进行增量训练，剩下的一组作

为测试集数据，每个集合中的恶意和正常连接的比例都大约为 1:1，并从经验测试集，也就是黑名单库中随机抽取 500 个链接和第十组数据进行合并，组成含有 800 条数据的新的测试集 *T*，这样，新的测试集中的新恶意样本、经验测试样本、新安全样本的比例为 1:3.3:1。具体的训练和检测验证过程分为以下几步进行:

Step 1：使用上一小节生成的 SVM 分类器假定为$f(a)$对新的测试样本集 *T* 进行恶意链接检测。

Step 2：将当前分类器$f(a)$利用 ASVM 在 t_1~ t_9 这 9 个新的训练子集上进行自适应学习，并生成$f'(a)_1$~$f'(a)_9$这 9 个新的子分类模型。

Step 3：利用这 9 个分类器分别对测试样本集 *T* 进行测试，并求所有的测试结果的均值作为最终的分类结果。

经过实验对比，9 个分类器的最终分类效果如表 5-8 所示：

表 5-7 ASVM **分类器实验结果对比**

分类器	$f'(a)_1$	$f'(a)_2$	$f'(a)_3$	$f'(a)_3$	$f'(a)_4$	$f'(a)_5$	$f'(a)_6$	$f'(a)_7$	$f'(a)_8$	$f'(a)_9$
准确率	0.919	0.921	0.919	0.899	0.926	0.94	0.929	0.920	0.935	0.931

最终的结果对比如表 5-9 所示：

表 5-8 SVM **和** ASVM **分类结果对比**

分类器	$f(a)$	$f'(a)$
准确率	0.914	0.921

其中$f'(a)$是$f'(a)_1$~$f'(a)_9$这 9 个新的子分类模型分类结果的均值。

从以上两张表可以看出，新的子分类器的预测准确率基本都在[0.91，0.94]这个区间范围内，其中，88.88%的测试结果都在 0.92 以上。和标准 SVM 分类结果的 0.914 相比，自适应 SVM 在对新的恶意链接样本的检测方面准确率存在很小的提升。

5.4.3.2 DSCD 模块验证

这一小节，我们主要利用实际存在脚本注入的页面针对我们的 DOM 结构改变检测算法来进行验证。我们选取以下链接进行验证：

https://idol001.com/news/6761/detail/590fcfb37a1173b56a8b47b6

具体的步骤为：

Step1：将该链接利用 WebView 引擎进行渲染解析，得到的页面结构如图 5-4 所示：

图 5-4 WebView 引擎解析渲染后的 DOM 文档模型

Step2：DSCD 算法在内存内生成该页面的内存快照，并开始进入循环，通过同层节点比对实时监控 DOM 结构。

Step3：如图 5-4 所示，10 秒后钓鱼页面通过 iframe 加载，WebView 引擎更新 DOM 树，更新后的页面结构如图 5-5 所示：

图 5-5 更新后的 DOM 文档结构

可以发现，相比于原文档，新的文档树被插入了 iframe 框架，图 5-4 中箭头标识的代码就是其要跳转的钓鱼链接。

Step4：我们查看系统控制台输出的 log 日志，发现 10 秒后新插入的 DOM 节点详细信息被打印出来，同时，数据库中维护的待检测暗链接表中成功插入了提取出来的暗链接，如图 5-5 所示：

图 5-6 DSCD 模块抽取出的新增 DOM 节点信息和暗链接信息

5.5 本章小结

本章主要通过具体的实验验证，来对本文的几种排重算法、分类算法以及页面结构检测算法通过一定的指标进行实验验证。通过实验对比，我们发现

（1）针对数据排重来看，布隆过滤器可以针对小规模的数据进行细粒度的筛选，MSHASH 更适合与对大规模数据通过降维和调节分桶粒来快速进行排重处理。而 MSHASH-BF 则兼顾两者的优点，通过粗粒度筛选后细粒度过滤，以降低少许速度为代价，换取排重效果的提升。

（2）本章节的对比试验充分验证了 SVM 算法在恶意链检测方面运用可行性和优势以及其改进算法 ASVM 相比于标准 SVM 确实可以提升一定的预测准确率，但提升升效果不是特别的明显，这很可能是由于本实验选取的样本数据规模不大、维度不高、属性特征过于离散，导致无法体现出 ASVM 的优势。这一块可以留待后续研究和证明。

（3）实验中，DSCD 成功将通过 iframe 懒加载的暗链接进行提取，证明了该方法的可行性。但是针对实际业务中的大规模并发数据，该算法的可靠性、检测速度等重要指标都还未经验证。算法依旧存在可以优化的细节。

6 结语

6.1 本文的研究工作

本文通过对当前互联网环境下存在的恶意链接泛滥的问题进行介绍，先调研了目前国内外专家和网络安全工程师在恶意链接检测领域各自研究成果和应用方案以及各方案的优势和缺陷，然后研究了恶意链接的主要攻击形式和核心技术，介绍了 JavaScript 脚本的发展、运行原理、语言特色、以及被用来进行恶意链接攻击的安全漏洞。同时，本文学习和了解了图像识别等领域常用的几种分类学习算法的原理、优点以及缺点后将其应用于恶意链接的系统设计中。

基于以上的准备工作，本文设计了包括数据排重、黑/白名单匹配、SVM 分类学习、DOM 结构更新检测几大模块的恶意链接检测系统，详细介绍了系统的设计理念、系统架构、系统的整体流程、各个模块具体的工作流程和实现原理。本文希望通过一套检测系统，能够有效的提高恶意链接检测的准确率，提升检测效率，降低误报率。

最后，本文在第 5 章节的做了一系列的效果对比试验，利用实验数据来验证系统中运用的各种算法在实际运行时是否符合预期。

本文的创新工作主要有以下几点:

（1）系统调研和学习了恶意链接的由来、常见的攻击形式，分析了其在当前互联网背景下的造成问题的严重性。通过调研和学习目前国内外安全领域的研究者和工作人员提出的理论和实际应用方案的原理及优缺点，在借鉴这些成果的基础上进行了一定的创新和改进，提出了利用黑/白名单库过滤、支持向量机分类学习算法和 DOM 结构改变监测算法进行分类预测的方案。

（2）在数据排重方面分析了常用的布隆过滤器的优缺点，并在借鉴了 SimHash 算法的基础上做了改进优化，提出了 MSHASH 方案，并且将其和布隆过滤器结合，进一步优化为 MSHASH-BF 方法，并经过实验验证了其在保证计算速度的前提下提升了过滤效果。

（3）提出了通过实时对 DOM 文档树进行同层节点对比监测页面 DOM 结构变化的 DSCD 方案并用实际的案例验证通过。该方案可以将页面中存在的暗链接进行提取存放如恶意链接测试样本集。

（4）详细设计和实现了恶意链接检测系统的架构和工作流程，介绍了各个模块的实现原理。

6.2 本文存在的不足和未来的改进

本文提出的恶意链接检测系统(SD-MLDS)经过实验验证，利用训练样本集构建恶意链接分类模型进行恶意链接检测，并利用 DSCD 算法提取出页面隐藏的暗链接，提高了对恶意链接的识别率和准确度。但是，本论文依旧存在一些问题以及可以改进的地方，具体如下：

（1）本系统目前只经过了实验数据验证，取得了部分阶段性的成果，但是还未经过具体的实际线上业务检测。

（2）基于黑/白名单过滤的规则太过简单，可以结合特征匹配等技术进一步提高该模块的检测效果，减少后续流程的开销。

（3）本文针对样本集预先限定了 36 个固定的属性特征向量来进行分类器的训练学习，然后在实际的业务场景中，恶意链接的形式远比本文分析的复杂，而且具有更强的隐蔽性和演变能力和反检测能力。针对这种情况，如何更好的选取训练集属性特征，并实现属性特征的自适应，这将是本文后续需要继续进行的工作。

（4）SD-MLDS 目前只能针对链接地址进行预测分类，但是，恶意链接仅仅只是复杂的网络安全问题中的一种，如何增加对恶意脚本、DNS 劫持等攻击行为的预防和检测拦截，提升系统的适用范围，留待后续的进一步研究设计。

（5）此外，由于 ASVM 需要不断更新训练集并重新训练分类器，那么系统中的样本集必将随着时间的推移而增大，这会导致系统额外的存储开销、计算开销、更新时常越来越大。这一问题，需要在后续的工作中进行优化和解决。

参考文献

[1] 唐发明, 王仲东, 陈绵云. 支持向量机多类分类算法研究[J]. 控制與决策, 2005, 20(7): 746-749.

[2] 丁世飞, 齐丙娟, 谭红艳. 支持向量机理论与算法研究综述[J]. 电子科技大学学报, 2011, 40(1): 2-10.

[3] Cao L J, Tay F E H. Support vector machine with adaptive parameters in financial time series forecasting[J]. IEEE Transactions on neural networks, 2003, 14(6): 1506-1518.

[4] 田苗苗. 数据挖掘之决策树方法概述[J]. 长春大学学报, 2004, 14(6): 48-51.

[5] World Wide Web Consortium. Document Object Model (DOM) Level 3 Core Specification[J]. 2004.

[6] Koetsier J. Evaluation of JavaScript frame-works for the development of a web-based user interface for Vampires[J]. 2016.

[7] 单长虹. 杀毒软件编程精华——特征码扫描技术[J]. 黑客防线, 2004 (03S): 48-49.

[8] Chabchoub Y, Fricker C, Mohamed H. Analysis of a bloom filter algorithm via the supermarket model[C]//Teletraffic Congress, 2009. ITC 21 2009. 21st International. IEEE, 2009: 1-8.

[9] Sadowski C, Levin G. SimHash: Hash-based similarity detection[J]. 2014-04-10]. www. Google code. com/sun/trunk/paper/Sim Hash with Bib. pdf, 2007.

[10] 李洋, 刘飚, 封化民. 基于机器学习的网页恶意代码检测方法[J]. 北京电子科技学院学报, 2013, 20(4): 36-40.

[11] 徐青. JavaScript 恶意代码检测技术研究[D]. 西南交通大学, 2014.

[12] 成亦陈, 黄淑华. 恶意短链接欺骗的防护对策研究[J]. 信息网络安全, 2013 (7): 0-0.

[13] 王维光. 基于分类算法的恶意网页检测技术研究[D]. 北京邮电大学, 2015.

[14] Sultana S, Bertino E, Shehab M. A provenance based mechanism to identify malicious packet dropping adversaries in sensor networks[C]//Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference on. IEEE, 2011: 332-338.

[15] Lawton G. Web 2.0 creates security challenges[J]. Computer, 2007, 40(10).

[16] 冯晓冬, 宋丽, 薄明霞, 等. 恶意程序监控系统在移动互联网安全防护中的应用[J]. 电信技术, 2013 (5): 45-47.

[17] 唐发明, 王仲东, 陈绵云. 支持向量机多类分类算法研究[J]. 控制與决策, 2005, 20(7): 746-749.

[18] Liu Y, Yang J, Liu M. Recognition of QR Code with mobile phones[C]//Control and Decision Conference, 2008. CCDC 2008. Chinese. IEEE, 2008: 203-206.

[19] Durkin J, 蔡竞峰, 蔡自兴. 决策树技术及其当前研究方向[J]. 控制工程, 2005, 12(1): 15-18.

[20] 张立彬, 张其前. 基于分类回归树 (CART) 方法的统计解析模型的应用与研究[J]. 浙江工业大学学报, 2002, 30(4): 315-318.

[21] Lin X, Zavarsky P, Ruhl R, et al. Threat modeling for csrf attacks[C]//Computational Science and Engineering, 2009. CSE'09. International Conference on. IEEE, 2009, 3: 486-491.
[22] 周佩颖. 恶意的 URL 捕获分析系统[D]. 电子科技大学, 2010.
[23] Cockburn A, Greenberg S, McKenzie B, et al. WebView: A graphical aid for revisiting Web pages[C]//Proceedings of the OZCHI. 1999, 99: 15-22.
[24] 张耀春. Vue.js 权威指南.北京: 电子工业出版社, 2016:312-314.
[25] Forstall S, Chaudhri I. Webview applications: U.S. Patent Application 11/145,560[P]. 2005-6-3.
[26] 林佳华, 杨永, 任伟. QR 二维码的攻击方法与防御措施[J]. 信息网络安全, 2013, 5: 015.
[27] 席晔文, 杨金民. 基于双布鲁姆过滤器的数据排重技术[J]. 计算机工程与应用, 2014 (23): 198-202.
[28] Pipe J G. Reconstructing MR images from under sampled data: Data‐weighting considerations[J]. Magnetic Resonance in Medicine, 2000, 43(6): 867-875.
[29] 孙飞. 基于 DOM 节点文本密度的网页核心块抽取算法研究[D]. 北京: 北京理工大学, 2011.
[30] 张立彬, 张其前. 基于分类回归树 (CART) 方法的统计解析模型的应用与研究[J]. 浙江工业大学学报, 2002, 30(4): 315-318.
[31] Yang J, Yan R, Hauptmann A G. Cross-domain video concept detection using adaptive svms[C]//Proceedings of the 15th ACM international conference on Multimedia. ACM, 2007: 188-197.
[32] Vapnik V N. An overview of statistical learning theory[J]. IEEE transactions on neural networks, 1999, 10(5): 988-999.
[33] 黄歆, 桑楠. 基于 DOM 树和递归 XY 分割算法的 Zone 树模型[J]. 计算机工程, 2009, 35(5): 53-55.
[34] 360.关于中国学术期刊标准化数据库系统工程的进展[EB/OL]. http://zt.360.cn/1101061855.php?dtid=1101062514&did=490278985, 2017-02-15/2017‐04‐05.
[35] 程杰仁, 殷建平, 刘运, 等. 蜜罐及蜜网技术研究进展[J]. 计算机研究与发展, 2008 (z1): 375-378.
[36] 宿洁, 袁军鹏. 防火墙技术及其进展[J]. 计算机工程与应用, 2004, 40(9): 147-149.
[37] Ursini F, Naty S, Grembiale R D. Fibromyalgia and obesity: the hidden link[J]. Rheumatology international, 2011, 31(11): 1403-1408.
[38] 熊斌, 晏继连. Web 中的异步加载应用[J]. 江西冶金, 2015, 35(4): 47-48.
[39] 古开元, 周安民. 跨站脚本攻击原理与防范[J]. 网络安全技术与应用, 2005 (12): 19-21.
[40] Bhargrava K, Brewer D, Li K. A study of URL redirection indicating spam[J]. CEAS (July 2009), 2009.
[41] 杨军, 刘艳, 杜彦蕊. 关于二维码的研究和应用[J]. 应用科技, 2002, 29(11).
[42] Quinlan J R. C4. 5: programs for machine learning[M]. Elsevier, 2014.

[43] 蒋良孝. 朴素贝叶斯分类器及其改进算法研究[J]. 博士论文, 中国地质大学, 武汉, 2009.
[44] Angluin D. Queries and concept learning[J]. Machine learning, 1988, 2(4): 319-342.
[45] 王永梅, 胡学钢. 决策树中 ID3 算法的研究[J]. 安徽大学学报: 自然科学版, 2011, 35(3): 71-75.
[46] Hou J, Zhang Y. Effectively finding relevant web pages from linkage information[J]. IEEE Transactions on Knowledge and Data Engineering, 2003, 15(4): 940-951.

MLA

致谢

时光飞逝，随着母校校庆的日子越来越近，我的三年研究生学习时光也即将结束。在完成本论文的过程中，三年来在学校中工作和生活过的日日夜夜都一遍遍的在眼前浮现，俨然是一份美好的收获和回忆。

首先要感谢我的导师卢明欣副教授。在我三年的研究学习期间，学术方面卢老师以严谨的学术态度、深厚的学术素养对我进行教导和培养，在我撰写学位论文的过程中，卢老师从本文的选题、理论方法的选择、初稿的修订到最终论文的完成提出了大量的宝贵意见，付出了无数的心血，帮助我最终完成本文。同时，在生活方面，卢老师也给予了我许多关心和帮助。这些都对我今后的人生有重要的影响。在此，我谨向卢老师表达我最诚挚的祝福。

同时还要感谢南京大学信息管理学院对我三年的细心培养，感谢学院每一位老师在工作和生活上给予的照顾和指导，感谢所有同学对我的信任与帮助。

此外，我还要特别感谢我的父母，他们用自己不求回报的付出将我抚养成人，支持我完成学业，并在我面对任何坎坷时给予最大限度的支持和鼓励，这份恩情无以回报。

最后再一次感谢南京大学对我教育和培养。

刘力铭

2017 年 5 月于南京大学仙林校区