

中图分类号：TP391.4

秘密 ☆ 永久

论文编号：10006SY1706404

北京航空航天大学 硕士学位论文

恶意网络爬虫检测与对抗技术的研究及实现

作者姓名 张浩凌

学科专业 网络空间安全

指导老师 李舟军 教授

何跃鹰 处长

培养院系 计算机学院

Research and implementation of malicious web crawler detection and confrontation technology

A Dissertation Submitted for the Degree of Master

Candidate : Zhang Haoling

**Supervisors: Prof. Li Zhoujun
He Yueying**

School of Computer Science and Engineering

Beihang University, Beijing, China

中图分类号：TP391.4

秘密 ☆ 永久

论文编号：10006SY1706404

硕 士 学 位 论 文

恶意网络爬虫检测与对抗技术的研究及实现

作者姓名	张浩凌	申请学位级别	工学硕士
指导老师姓名	李舟军	职 称	教授
学科专业	网络空间安全	研究方向	网络空间安全
学习时间自	2017 年 09 月 01 日	起至	2020 年 01 月 31 日止
论文提交日期	2019 年 11 月 25 日	论文答辩日期	年 月 日
学位授予单位	北京航空航天大学	学位授予日期	年 月 日

关于学位论文的独创性声明

本人郑重声明：所呈交的论文是本人在指导教师指导下独立进行研究工作所取得的成果，论文中有关资料和数据是实事求是的。尽我所知，除文中已经加以标注和致谢外，本论文不包含其他人已经发表或撰写的研究成果，也不包含本人或他人为获得北京航空航天大学或其它教育机构的学位或学历证书而使用过的材料。与我一同工作的同志对研究所做的任何贡献均已在论文中作出了明确的说明。

若有不实之处，本人愿意承担相关法律责任。

学位论文作者签名：_____ 日期：_____ 年 _____ 月 _____ 日

学位论文使用授权

本人完全同意北京航空航天大学有权使用本学位论文（包括但不限于其印刷版和电子版），使用方式包括但不限于：保留学位论文，按规定向国家有关部门（机构）送交学位论文，以学术交流为目的赠送和交换学位论文，允许学位论文被查阅、借阅和复印，将学位论文的全部或部分内容编入有关数据库进行检索，采用影印、缩印或其他复制手段保存学位论文。

保密学位论文在解密后的使用授权同上。

学位论文作者签名：_____ 日期：_____ 年 _____ 月 _____ 日

指导教师签名：_____ 日期：_____ 年 _____ 月 _____ 日

摘 要

随着各类基于大数据和 AI 的应用的兴起，能够快速廉价地获取大量有效数据的能力，成为互联网时代企业和个人竞争力的体现。因此，网络爬虫在数据收集收集方面的重要性逐渐凸显出来。但是，恶意爬虫同样给互联网用户和互联网服务提供者带来巨大的困扰，这些爬虫或者多线程高并发耗尽服务器的带宽和计算资源 [1]，或者爬取个人敏感信息、高价值的商业数据用于不法用途。

因此，构建一个成熟有效的反爬虫系统，成为一个亟待解决的问题。但遗憾的是，传统的反爬虫方式过于保守和被动，在漏检率和误检率居高不下的情况下，往往只能通过单一的反制手段来限制爬虫（如 IP 封锁，访问频率限制，虚假数据等），在反爬虫的战争中收效甚微。为此，本文提出了一种新型的反爬虫的系统，融合爬虫检测技术，爬虫行为分析和溯源技术，并通过动态符号执行、模糊测试以及污点分析等二进制漏洞挖掘方法，挖掘爬虫使用的框架、处理脚本，无头浏览器驱动程序驱动程序的漏洞，并通过返回恶意的攻击载荷，对运行恶意爬虫的主机进行反向攻击，最终诱使恶意爬虫进程奔溃，甚至可能获取到恶意爬虫主机上的敏感数据以及系统权限。

此外，本文将遵循上述的设计思路，实现该反爬虫系统的原型系统 **Crawler-Net**(爬虫网)，并将 **Crawler-Net** 部署在模拟的业务系统上，使用具有不同请求策略的爬虫流量混合正常的用户访问进行测试，从漏检率、误检率、系统性能损失等多方面的指标来评估反爬虫系统的性能。

关键字：恶意爬虫，反爬虫机制，无头浏览器，爬虫检测，爬虫溯源，动态符号执行，模糊测试

Abstract

With the flourish of the Applications based on the Big Data and Artificial Intelligence, it has aroused our attention in how to collect numerous valid data rapidly at the least cost, which could be regarded as an aspect of competitive competence both for the individuals and the companies. And thus, the power of the crawlers in collecting data has been addressed. However, there are plenty of malicious crawlers filling in the cyberspace, try to deplete the servers' resource with endless requests concurrently, or theft the sensitive individual's information or commercial data for illegal use.

Therefore, we are supposed to build up a feasible, active and robust anti-crawling system to stop those malicious crawlers, while most of the contemporary anti-crawling systems are using passive strategies. Those anti-crawling systems could only harness single mechanism to block the crawlers(such as IP blocking, limit the frequency of requests, fake data), and that does not seem to be effective in most cases. In this paper, we propose an anti-crawling system, which combines the crawler detection with the analysis of crawler's behaviors, then utilizes the dynamic symbolic execution, fuzzing and dynamic taint analysis , to excavate the vulnerabilities of the frameworks, the handling scripts and the headless web-driver binaries of the crawlers. Ultimately, malicious payloads would be generated to feed the crawler when it crawls our web pages, leading it to crash down or even obtain the sensitive data as well as the system privileges from the crawler's host.

In addition, at the end of the paper, I would build up the prototype of the anti-crawling system called "Crawler Net" according to the technologies and mechanisms mentioned before, with the deployment to the web application in a simulated production environment. Furthermore, several tests would be conducted with the requests generated by various types of crawlers and the requests from normal user to testify its ability to block the crawlers and the performance once deployed.

Key words: malicious crawler, anti-crawling system, headless browser, crawler detection, crawler trace, dynamic symbolic execution, fuzzing test

目 录

第一章 绪论	1
1.1 研究背景及意义	1
1.2 国内外研究现状	2
1.3 研究目标和内容	4
1.3.1 对抗样本下 session 分类方法的研究	4
1.3.2 基于 session 粒度的爬虫识别与检测	4
1.3.3 基于隐式浏览行为的爬虫识别	5
1.3.4 针对爬虫 webdriver 的自动化漏洞挖掘技术	5
1.4 本文的组织结构	5
第二章 主流爬虫技术的研究	7
2.1 爬虫分类	7
2.2 爬虫调度策略与爬行策略	7
2.3 无头浏览器技术	9
第三章 恶意爬虫检测技术研究	11
3.1 爬虫样本与训练数据	11
3.2 单粒度模式检测规则	13
3.3 session 粒度模式检测	15
3.3.1 静态 session 分类	15
3.3.2 动态 session 分类	16
第四章 说明	20
4.1 宏包使用	20
4.2 选项设置	22
4.3 章节撰写	22
4.4 注意事项	23
4.5 ToDo	23

4.6 意见及问题反馈	24
第五章 示例	25
5.1 参考文献引用	25
5.1.1 数字标注	25
5.1.2 数字标注-上标形式	25
5.1.3 著者-出版年制标	26
5.1.4 其他形式的标注	26
5.2 浮动体	27
5.3 算法环境	27
5.3.1 三线表	27
5.4 长表格	29
5.5 插图	30
5.6 数学环境	31
5.6.1 数学符号	31
5.6.2 定理、引理和证明	31
5.6.3 自定义	33
结 论	35
参考文献	36
附 录	36
攻读硕士学位期间取得的学术成果	37
致 谢	38
作者简介	39

图 清 单

图 1	selenium 架构图	10
图 2	清华大学某网站爬虫组成	11
图 3	清华大学某网站访问 session 长度分布图	12
图 4	user-agent 关键字	13
图 5	curl 与 chrome 的访问请求对比	14
图 6	curl 与 chrome 的访问请求对比	16
图 7	测试图片第二行题注	30

表 清 单

表 1	已有研究成果比较	3
表 2	静态 session 分类效果比较	16
表 3	不同浏览器指纹比较	19
表 4	表的标题	28
表 5	让我们看看一个长标题长什么样。还不够长? 那我再多写一点。还是不够 长? 那我再多写一点点。OK, 就是长这样的!	28
表 6	长表格演示	29

主要符号表

E 能量

m 质量

c 光速

P 概率

T 时间

v 速度

第一章 绪论

1.1 研究背景及意义

机器学习, 数据挖掘等大量数据依赖型的技术在高速发展的同时, 对于相关数据的质和量都提出了更高的要求。网络爬虫, 又称网络蜘蛛或者网络机器人, 在这样一个数据消费的时代, 扮演着数据搬运者和传递者的角色。在其运行的生命周期中, 往往会按照开发者预设的规则, 爬取指定的 URL 地址或者 URL 地址列表, 并将获取到的数据预处理成标准化的格式 [2]。

普通的网络爬虫并没有危害, 相反, 搜索引擎存储数据的来源都是基于大量分布式网络爬虫爬取得到的结果, 网络爬虫在数据传递过程中起到至关重要的重要。但是多线程高并发的失控爬虫, 针对网站隐私数据的窃密爬虫, 以及不遵守爬虫道德规范的恶意爬虫, 都对网络空间健康的生态环境提出了巨大的挑战。

爬虫失控往往是具有多线程操作的通用型爬虫, 未能控制爬行时间间隔, 或者因为未添加地址环回检测的处理逻辑, 在处理特殊的地址链接时陷入死循环中。失控的爬虫通常给网站的性能资源和带宽资源带来巨大的消耗, 甚至会影响正常人类用户的体验, 这样的爬虫对网站的影响相当于是 DDOS 攻击 [1]。每年的三月份, 是失控爬虫的高发期, 原因正是因为大量的硕士在写论文时会爬取网站数据用于数据挖掘或者机器学习。

此外, 部分互联网公司会爬取其他同行的优质数据用于商业用途, 在给其竞争对手带来经济效益的损失的同时, 还会促进产业内部恶性竞争的循环。例如, 马蜂窝网站的用户评论数据涉嫌造假事件。甚至在一些情况下, 恶意爬虫甚至会爬取敏感个人信息用于不法用途, 例如某大数据公司非法爬取个人信息被起诉一案。

某些由黑客或者 APT 组织控制的爬虫, 在爬取某些 CMS 系统或者 web 中间件的版本信息后, 会使用相应的攻击向量攻击脆弱主机 [6], 给网站服务提供者及使用者造成巨大的损失。网络空间一直是爬虫与反爬虫战斗的前线, 随着反反爬虫技术的不断迭代更新, 传统的静态、单一、被动与非实时的反爬虫技术难以与之对抗, 或者又因为部署成本和部署带来的性能损失而被束之高阁。

1.2 国内外研究现状

在数据需求不断增加的大数据时代，爬虫技术的发展也日新月异。与此同时，传统单一静态的爬虫识别技术已经无法满足现阶段的需求，研究相关领域的学者也一直在为反爬虫领域提供新的技术和新的思路。

Guo W 等人首先针对传统与单个 http 请求进行处理的爬虫识别算法提出新的改进，在他们的文章中，他们首先使用了 session 粒度的爬虫识别算法，重点关注人类访问 session 与爬虫访问 session 中对应的 url 请求资源类型（样式表，html，图片等）比例不同的特性，并对采集得到的相应的日志进行非实时的线下处理，提取出相关特征作为分类依据。这是最早的基于 session 的反爬虫机制。

Derek Doran 在他们的文章中使用了和 Guo W 等人类似的方法，他们在使用 url 请求资源类型比例作为重要特征的同时，引入了离散马尔科夫链的概率模型，并使用该模型得出的 log 概率来判定访问时来自于人类还是爬虫。但是总体而言，该文章提出的模型并没有过多的创新，而且马尔科夫链的概率模型的计算过程，需要消耗大量的计算资源，不能够应用在实时的爬虫检测上。

与此同时，为了将爬虫识别模型应用到真实的业务场景中，而不仅仅作为一种离线的验证算法。Andoena Balla 等人提出了实时的爬虫识别算法。在他们的文章中，他们还引入了如下的 session 粒度下的特征：（1）head request 的百分比（2）2xx 返回的比例（3）3xx 返回的比例（4）页面资源的访问比例（5）夜间访问的比例（6）访问两个页面之间的平均时间（7）其他二进制文件请求的比例。

Andoena Balla 的对 session 粒度使用了比较完备的特征，为后来基于 session 粒度的爬虫识别的相关研究，提供了有价值的参考。但对于如何进行 session 的分类以及如何处理 session 过长的问题，该文章并没有提供合格的解决方案。Yi Liu 在他们的文章中提出了一种解决 session 过长问题的方法。他们采用了滑动窗口的机制，对每一次处理的 http 请求做了相关的限制，对在最大程度上保证了处理的 http 请求的相关性，并在一个窗口内使用 SVM 来区分普通用户和爬虫。此外，他们还以他们的模型为基础，实现了一整套支持实时爬虫检测的系统。但是该文章在 session 粒度的特征方面，并没有充分利用 session 中的包含的信息。

Shengye Wan 在他的文章中综合了现有的反爬虫技术，提出一个名为 PathMarker 的

模型。**PathMarker** 会将 url 地址信息以及当前访问的用户信息加密, 并替换掉原有的 url 地址。由此标注每个请求所对应的 session, 并利用之前的研究中使用的 session 特征, 来区分爬虫和人类。这种方案可以在很长的时间窗口内持续地追踪爬虫在某个网站的爬行轨迹, 可以用于分析爬虫的爬行目标和爬行策略, 并且在某种程度上可以追踪使用分布式 ip 池的爬虫群。**PathMarker** 在 session 分类良好的情况下能够产生较好的爬虫检测效果, 但是其缺点也很明显, 需要修改所有返回请求中的所有链接地址, 在真实的生产环境下难以提供灵活的部署方案。此外, 一旦访问的爬虫使用对抗性的爬虫策略, **PathMarker** 的 session 分类效果将不再准确, 从而影响最终的爬虫识别效果。

总体而言, 现有的反爬虫技术主要注重于爬虫识别技术的发展, 发现可疑爬虫后的阻断方法, 往往是单一的阻断或者使用 **captcha** 机制进行验证, 无法对恶意爬虫的作者起到威慑的作用。在爬虫识别技术的主流技术中, 在单粒度上识别的检测技术, 一旦遇到采用字段变异的爬虫, 便无法发挥应有的作用。而在 session 粒度上识别的检测技术, 在实时性检测上, 往往都有较大的性能消耗。除此之外, 大部分的爬虫识别模型都没有考虑到爬虫可能采用的反反爬虫手段, 对于一些有着特殊对抗策略的爬虫, 其识别效果将大打折扣。

表 1 已有研究成果比较

论文题目	实时性	静态信息利用	动态信息利用	检测粒度	性能损耗	爬虫对抗
Protecting Web Contents Against Persistent Crawlers	Y	refer, user agent, cookies	N	session	高	N
Web robot detection techniques based on statistics of their requested URL resources	N	user agent, URL pattern	N	单粒度/session	高	N
RESEARCH ON AN ANTI-CRAWLING MECHANISM AND KEY ALGORITHM BASED ON SLIDING TIME WINDOW	Y	N	N	session	中	N
Detecting web robots using resource request patterns	N	N	N	session	高	N
Real-time Web Crawler Detection	Y	N	N	session	中	N
Our paper (Crawler-Net)	Y	user agent, http headers value, http headers key order	Y	单粒度/session	中	Y

1.3 研究目标和内容

本文的研究目标是提出并实现一种新型的针对恶意爬虫的检测和攻击的对抗性技术，用于在爬虫实时访问阶段，对爬虫进行实时监测与阻断，并针对识别出的、具有恶意探测行为和攻击行为的爬虫，在检测其相应的 `webdriver` 的类型和版本后，生成相对应的攻击载荷。

并在该技术的基础上，实现恶意爬虫识别与攻击系统。该系统的目标功能为：1) 捕获到目标网站的所有访问请求，并实时地将其按照 `session` 粒度进行分类；2) 通过不同粒度下的数据分析，检测和识别 `HTTP` 请求中出现的爬虫；3) 根据设定阈值，从爬虫 `session` 中筛选出恶意行为的爬虫；4) 利用事先生成的攻击载荷攻击恶意爬虫，并收集相应攻击返回。

基于以上研究目标，本文的主要研究内容包括以下几个部分：

1.3.1 对抗样本下 `session` 分类方法的研究

在普通的爬虫和正常的人类访问行为下，针对给定的 `http request` 序列进行 `session` 分类并不是一个困难的问题。通常我们在某一特定时间段内，将同一 `ip` 来源或者使用同一 `cookie` 的 `http request` 分类为同一个 `session`，但是在存在对抗行为的恶意爬虫面前，这样粗糙的分类方式通常难以获得令人信服的分类结果。参考利用 `http` 请求的静态特征以及利用 `javascript` 执行得到的动态特征，以此得到一个更加精确的 `session` 分类结果，是我们研究关注的重要内容之一。此外，针对一个真实生产环境下收集的实时数据，如何实现实时的 `session` 分类处理，以及各类参数和阈值（如 `session` 分类的时间长度和序列长度）的合理设定，也是我们研究的一部分。

1.3.2 基于 `session` 粒度的爬虫识别与检测

传统基于单个 `http request` 的爬虫识别方法，仅仅基于一些 `http` 字段的静态特征，很容易使用一些额外的工具和算法对 `http` 字段进行变异，从而逃避传统反爬虫程序的检测。因此，我们考虑到从 `session` 中额外提取爬虫的特征，并参考原先的静态特征，共同作用于爬虫识别。目前已经有一些论文讨论到基于 `session` 的爬虫识别，并给出了相关了 `session` 粒度下的爬虫特征 [4] [6] [7]，但是这些文章提出的一些特征存在一些问题：一部分特征实时检测的性能开销过大；一部分特征在数据集中并不显著，或者对恶意爬虫的支持不佳；因此，通过本次研究，我们还会得到一系列 `session` 的特征以及这些特

征的相关阈值设定。

1.3.3 基于隐式浏览行为的爬虫识别

正常的人类浏览行为下，除了访问频率和访问间隔于爬虫存在差异之外，还会存在一部分隐式的浏览行为 [9]。例如，在相关页面插入一些不在浏览器上渲染但是存在于 html 代码中的链接，在正常访问下不会被触发，但是遇到了爬虫的 html link parser，则可能会被访问，由此我们可以判定访问该链接的请求以及对应的 session 均是来自于爬虫的。如何能够利用人类隐式的浏览行为，来进行更为精细准确的爬虫识别，也是本研究的重点。

1.3.4 针对爬虫 webdriver 的自动化漏洞挖掘技术

一旦识别出恶意爬虫，在此基础使用 javascript 探测其 webdriver 版本以及可能存在的漏洞类型，并通过此漏洞生成相应的攻击载荷实现对爬虫的攻击，这是我们系统的最终的目的。如何通过已知的 webdriver 版本，来生成有效的攻击载荷，我们需要借助于已经成熟的自动化漏洞挖掘技术。虽然单个的 webdriver 程序体积很大，但是因为我们目标的 webdriver 往往是开源的。因此，本课题将研究在获得源代码的前提下，如何对程序功能进行分割或者采用一些其他的简化技术，预防可能出现的路径爆炸问题，并尽可能多地找到一些漏洞。

1.4 本文的组织结构

本文的组织结构如下：第一章为绪论，总体介绍爬虫技术，爬虫检测技术等相关概念，及研究恶意爬虫对抗技术的重要意义。并总结国内外工业界和学术界针对反爬虫技术的研究现状，并指出目前已经工具及分析方法的局限性，提出本文的研究目标和内容。

第二章为主流爬虫技术的研究部分，通过对已有的爬虫技术的研究，对主流的爬虫依据其特性进行分类。并概述目前爬虫技术中使用的基本的爬行策略以及针对现有反爬虫技术发展而来的伪装策略，分析进行爬虫识别过程中的主要挑战和难点。最后重点分析具有动态处理功能的爬虫底层使用的无头浏览器技术及该技术的特性，讨论了针对无头浏览器的对抗技术实施的可能性。

第三章为恶意爬虫检测技术的研究部分，爬虫的检测技术主要从三个方面入手：(1)

单粒度模式下的检测技术（2）基于 session 粒度的检测技术（3）基于隐式浏览行为的爬虫识别。其中 session 分类与特征提取是本文的核心研究内容之一。并在第三章的最后，介绍了我们的爬虫检测技术在三个不同数据集上的检测效果。

第四章为恶意爬虫对抗技术的研究部分。重点介绍了四种对抗技术：（1）资源耗尽型的对抗技术（2）进程 crash 类型的攻击技术（3）数据窃取类型的攻击技术（4）爬虫数据流追踪技术。爬虫检测技术产生的结果将作为生成对抗策略的依据，对抗策略将决定对抗技术的种类以及相应攻击载荷的生成。

第五章为恶意爬虫对抗系统的架构研究和实现部分。通过对整个对抗系统运行流程的分析，设计了由四部分组成的具有较强健壮性的恶意爬虫对抗系统，并具体地介绍四部分间协同合作的运作方式。

第六章为恶意爬虫对抗系统的评估和分析，将使用多策略爬虫工具测试整个系统的识别率和误报率，以及在高并发情况下的性能损失比率，并进一步分析产生这样结果可能的原因。

最后是整篇文章的结论部分，总结全文的工作，对本文仍然存在的不足之处提出了更为深远地思考，同时包含了对未来爬虫、反爬虫技术互相迭代的展望。

第二章 主流爬虫技术的研究

2.1 爬虫分类

网络爬虫大致可分为四种。通用网络爬虫，聚焦网络爬虫，增量式网络爬虫，深层网络爬虫 [2,7]。

- (1) 通用爬虫：通用网络爬虫又称全网爬虫 (Scalable Web Crawler)，主要的爬取对象是大型的门户网站、搜索引擎和大型的 Web 服务提供者。通常会从一个初始的 URL 地址开始，以其为爬取的种子，递归地搜索遍历该种子地址下的所有 URL 地址。
- (2) 聚焦式网络爬虫：聚焦网络爬虫 (Focused Crawler)，又称主题网络爬虫 (Topical Crawler)[8]，是指选择性地爬行那些与预先定义好的主题相关页面的网络爬虫。通常只需要爬取与其感兴趣的主题相关的页面，保存的页面内容数量少，更新快。
- (3) 增量网络爬虫：增量式网络爬虫 (Incremental Web Crawler) 将参照已下载的网页内容，采用增量更新的方式只爬取没有下载或者已经内容发生变化的页面。以较高的访问频次来保证爬取的内容的实时性，但是因其只请求和持久化增量的静态内容 [9]，所以请求的数据量较少。
- (4) 深层网络爬虫：web 页面按其存在方式一般分为表层网和深层网。表层网通常可以由搜索引擎检索，以超链接的形式能够到达，Deep Web 指的是不能直接检索，隐藏在表单之后的网页 [3]。深层网络爬虫 (Deep Web Crawler) 主要用于爬取普通爬虫无法爬取到的深层网络，通常是提交某些参数（如登录操作）后才能到达的页面 [10]。

2.2 爬虫调度策略与爬行策略

爬虫的目的是在短时间内尽可能获得最多的高质量数据。当前有五种评估页面质量的机制 [11]：Similarity，Backlink，Pagerank，Forwardlink 以及 Location。

而为了爬虫的爬取速度，爬虫往往以并行地方式运行，但也引入了如下的问题 [7]：

- (1) 重复性：线程增加的同时可能增加页面重复的几率

(2) 通信带宽损耗：分布式爬虫之间的同步通信损耗大量的带宽和计算资源

并行运行时，爬虫通常采取如下三种方式 [7]：

- (1) 独立运行：所有爬虫独立运行，互不影响，全部爬取完成后，再统一处理爬行资源。
- (2) 动态队列分配：使用统一的任务队列分配爬虫的爬取任务，每一次爬虫从任务队列中获取到的爬取任务，都是动态且随机的。
- 3 静态分配：在爬虫任务开始前，事先为所有的爬虫分派爬取的任务，在爬虫爬取过程中不再做额外的修改。

对于通用爬虫而言，常用的策略主要有深度优先爬行和广度优先爬行两种 [12]。

- (1) 深度优先：将链接深度从低到高排列，依次访问下一级网页链接直到不能深入为止。在完成一个分支网页的爬行任务后，返回上一个链接节点，进一步遍历其他链接。
- (2) 广度优先：按目录层次深浅来规划爬行的轨迹，从浅层的 web 目录开始爬行，低层次的目录爬行完毕后，再深入下一层继续爬行。可以有效的控制爬行的深度，避免无穷深层次分支的情况下无法走出陷阱。

聚焦爬虫策略的关键是对访问页面和链接的重要性进行评分，并以此作为页面爬取先后的顺序的参考依据，而不同算法计算出的页面重要性也有所不同，从而导致爬虫爬行轨迹的不同。

- (1) 基于内容的爬行策略：Bra 在他的论文中提出了 Fish-Search 算法 [13]，以页面内容与爬虫的聚焦主题的相关性作为度量标准。假设存在鱼群，鱼群按照深度优先的策略在爬行空间中巡游，相关度高的分支，鱼群的数量会增加；而相关度低的分支，鱼群的数量会降低。通过这样的机制来保证主要的爬虫资源和时间都能够聚焦在感兴趣的主题上。
- (2) 基于链接结构的爬行策略：Page-rank 算法 [14] 主要用于对搜索引擎搜索到的内容进行结果排序，也可以用于评价链接的重要性。在爬虫选取爬行链接时会优先选取 Page rank 较大的页面中的链接作为优先爬取的对象。
- (3) 基于增强学习的爬行策略：Rennie 和 McCallum 将增强学习 (reinforcement learning) 引入聚焦爬虫 [15]，他们试图通过机器学习找到一组策略，使得在该策略的激励

达到最优解。

- (4) 基于语境图的爬行策略: Diligenti 提出一种通过建立语境图来描述不同网页之间相关度的算法 [16]。用大量的数据训练出一个机器学习系统, 利用该系统计算不同页面到某一主题页面的距离, 以此作为确定访问顺序的参考。

2.3 无头浏览器技术

随着反爬虫技术的发展, 大部分的网站都通过开启 javascript 的检查来避免一些静态爬虫的爬取。在第一次访问网站的时候, 网站会返回一段经过混淆的 javascript, 该 javascript 执行完毕后会在用户的浏览器中设置相应的 access key, 并在之后所有向该网站发送的请求中添加该 access key。同时, 网站的服务器端将会拒绝所有不带有 access key 的所有请求。通过这种方式, web 服务提供者可以避免网络中 80% 的爬虫请求的骚扰。

为了对抗这种 javascript 检查的机制, 爬虫需要开启对于 javascript 脚本的支持。而传统的爬虫只有数据获取部分、数据解析的功能, 并没有集成 javascript 执行的功能, 因此需要借助于外部的 javascript 解释执行引擎。而无头浏览器 (headless browser) 作为一种成熟的解决方案, 被爬虫技术广泛地作为 javascript 的执行引擎。无头浏览器和普通浏览器最大的区别是无头浏览器在启动后不会生成可视的 UI 界面, 所有的 javascript 执行和画面渲染全部在内存中进行, 除此之外, 无头浏览器和普通的浏览器基本上没有差异。比较著名的 headless browser 有 chrome headless 以及 phantomjs, 而且目前的主流浏览器大都支持 headless 模式, 通过关闭 ui 和开放相应的操作接口来支持自动化。phantomjs 可以运行大部分主流的操作系统上, 使用 QtWebKit 作为后端, 可以支持各种各样的 web 标准, 包括但不限于: DOM 树解析, CSS 选择器, JSON 数据解析, CANCAS 渲染等。

对于具体的 headless browser 我们有着不同的选择, 而 selenium 自动化测试工具, 为我们与大部分的 headless browser 之间的交互提供了编程接口。在 selenium 升级到 2.0 之后 [31], 它重点发展了 selenium webdriver, 由以前的使用 selenium server 与浏览器通信到使用浏览器自身的 webdriver 与浏览器进行通信, 绕过了 javascript 沙箱, 提升了测试效率和对各个浏览器的兼容性。selenium 对外分开放了大量 API 接口, 可以使用各类编程语言控制测试过程。selenium 架构图如下图所示:

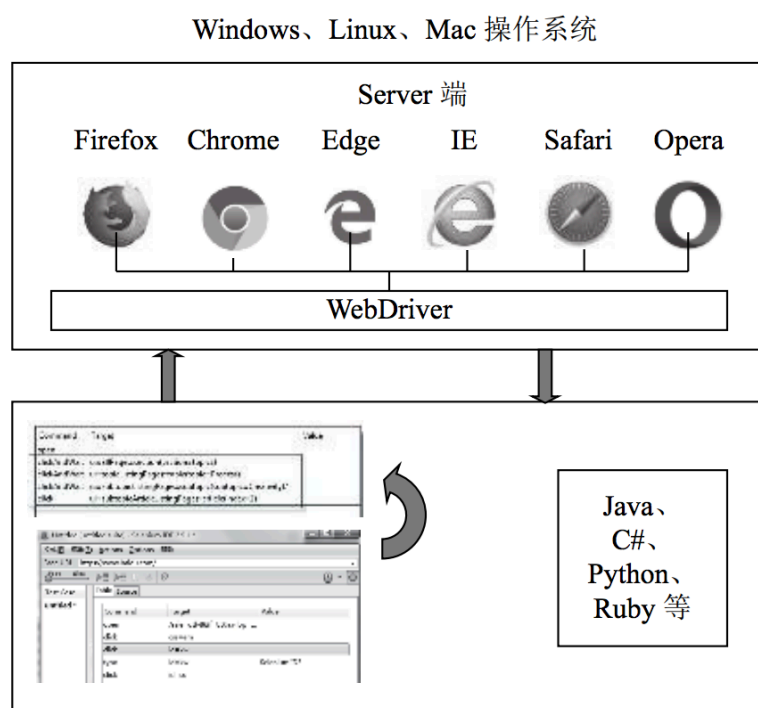


图 1 selenium 架构图

第三章 恶意爬虫检测技术研究

3.1 爬虫样本与训练数据

为了检验和评估不同的方案下，爬虫识别算法的检测率和效率，我们使用了三个不同的数据集进行评估。前两个数据集分别为：

- 清华大学某网站访问数据集（数据集 a）
- 360 天眼恶意攻击流量数据集（数据集 b）
- 多策略爬虫工具生成数据集（数据集 c）

两个数据集的量级均在在 10w+ 左右，其中数据集 a 记录的是清华大学某重要网站在某时间段内的所有请求。通过基于 User-agent 的传统方法针对数据集 a 进行清洗，我们发现基于 User-agent 的方法识别出请求中的 24.3% 为来自各类爬虫的请求，爬虫的组成成分主要是来自于各大搜索引擎的爬虫以及专门针对清华网站的爬虫，如下图所示。因为该数据集几乎不存在对抗样本，因此我们使用 ip 作为 session 分类的标准，对全部的 10w 个请求进行 session 分类，在去除一些超长和超短的 session 之后，我们可以得到了以下的 session 的长度分布。

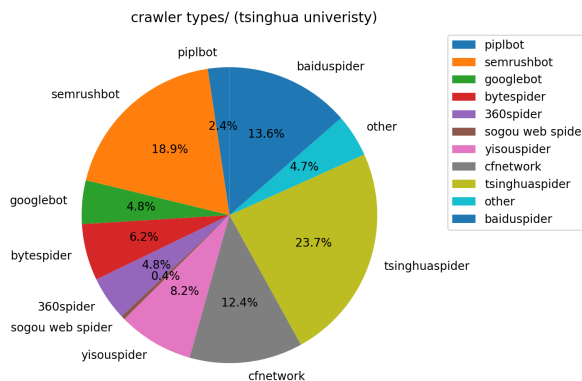


图 2 清华大学某网站爬虫组成

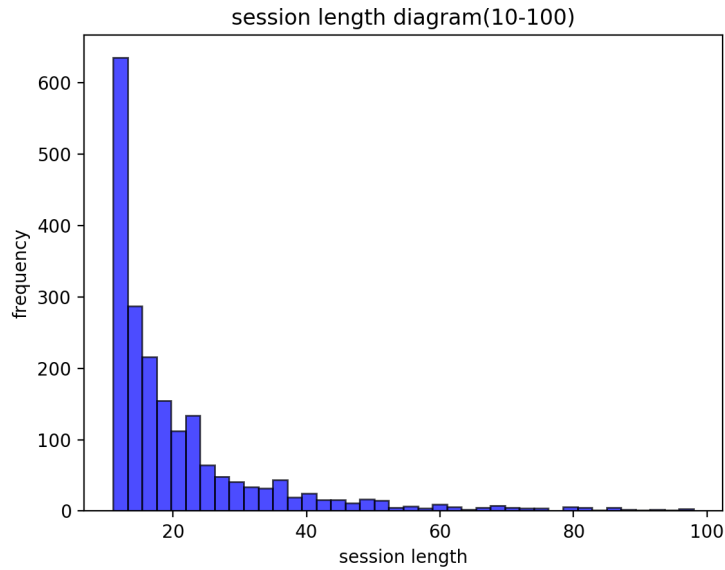


图 3 清华大学某网站访问 **session** 长度分布图

数据集 **b** 来自于 360 天眼捕获的恶意攻击的流量，这些请求数据绝大部分是互联网上的漏洞扫描爬虫发送的，带有明显的攻击载荷的特征。和传统的爬虫访问流量不同，这些恶意攻击的请求使用了大量的对抗性策略，例如分布式爬取以及 **http** 字段变异，通过这些手段，来干扰爬虫识别时的 **session** 分类。这对已有论文中，使用 **session** 粒度的爬虫检测方法提出了巨大的挑战。该数据集主要应用于验证本文中的 **session** 分类方法在遇到遇到对抗性策略的爬虫时，仍能够保持较高的 **session** 识别准确率。

但是，静态的访问数据并不能满足我们的需求，因为静态的数据中，缺乏爬虫与服务器之间的交互信息，而在真实情境下，服务器可以通过改变返回的页面中的 **URL** 地址，以及返回的 **javascript** 代码来影响甚至操纵爬虫的行为。因此，为了模拟真实网络环境中的爬虫请求，我们还设计了一系列的爬虫策略，并基于这些策略，生成了近千种不同类型的网络爬虫，我们称之为多策略爬虫工具。我们考虑到的爬虫特性有：

- 爬虫的目标数据
- 爬虫的请求速率
- 爬虫的遍历算法
- 爬虫的驱动程序
- 对抗性的爬虫策略

这些不同类型的爬虫将会产生大量普通的爬虫请求以及具有对抗性的爬虫请求，这

些爬虫请求在产生的过程中会记录 session id 用以检测 session 分类的效果，并且这些爬虫均具有能够执行服务器反馈的 javascript 的特性，因此可以用于收集不同 webdriver 下的动态指纹，这些爬虫产生请求将会作为本研究的第三个数据集（数据集 c）。

3.2 单粒度模式检测规则

单粒度模式检测指定是在不使用 session 信息以及动态信息的同时，仅仅利用单个 http 请求中所有的可用信息，来判定该请求是否源于爬虫的技术。该技术也是应用最广泛、性能消耗最少的爬虫检测方法。因为其对整个系统并没有什么性能损耗，因此在我们的模型和最终的系统中，我们仍然保留单粒度模式检测规则。

在我们的模型中，我们对如下内容进行检测：

- **User-agent 字段：**User-agent 字段主要是访问者的浏览器或者相应的 http 库标识当前访问者使用的 agent 的一个重要字段。大部分的普通爬虫，都会在其 user-agent 字段放置与该爬虫有关的信息，通过该信息，我们可以快速地筛选出这些爬虫。但是，由于 user-agent 字段是由用户发送的，用户可以随意修改该字段来达到隐蔽真实 user-agent 的目的，因此，该字段是不可信的。在我们的模型中，通过参考已有的爬虫数据库，我们采用如下的关键字来处理 user-agent 字段：

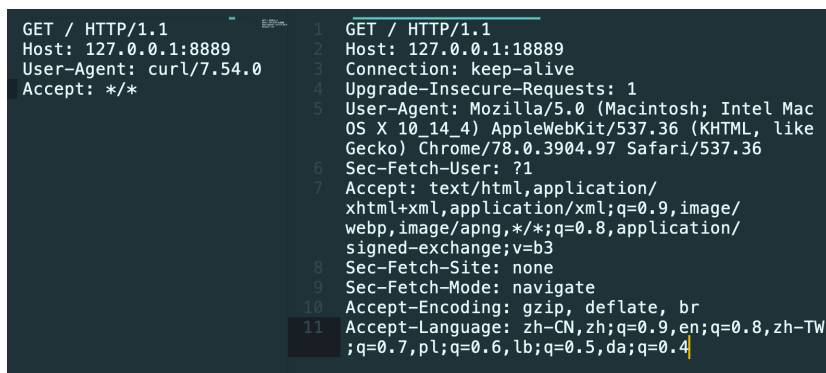
```
[ 'baiduspider', 'tsinghuaspider', 'yisouspider', '360spider', 'sogou-web-spider', 'bytespider', 'magibot', 'coccobot', 'semrushbot', 'piplbot', 'bingbot', 'googlebot', 'cliqzbot', 'exabot', 'ahrefsbot', 'yak', 'feeddemon', 'indy library', 'alexa toolbar', 'asktbftxtv', 'ahrefsbot', 'crawldaddy', 'coolpadwebkit', 'java', 'feedly', 'universalfeedparser', 'apachebench', 'microsoft url control', 'swiftbot', 'zmeu', 'obot', 'jaunty', 'python-urllib', 'lightdeckreports bot', 'yyspider', 'digext', 'httpclient', 'heritrix', 'easouspider', 'ezooms', 'headless', 'curl', 'wget', 'bot', 'spider', 'cfnetwork', 'python', 'crawler', 'go-http-client', 'ruby' ]
```

图 4 user-agent 关键字

- **Http 请求方法：**http 请求方法指的是浏览器等在访问 web 服务器时，于 http 请求头中设置的请求方法。常用的请求方法有：HEAD, GET, POST, PUT, OPTIONS 等，而正常情况下，浏览器所使用的请求方法只有 GET, POST，但是大量的爬虫为了爬取效率常常会使用 HEAD 方法，部分漏洞扫描器则会使用 PUT 方法来攻击一些有弱点的服务器（ps: 此处加上引用 iis put 漏洞 + tomcat put 漏洞）。因此，对 http 请求方法进行判断也能够识别出一部分的爬虫。
- **特殊的 URL 地址：**部分网站会在网站的 web 根目录下放置 robots.txt，该文件是

web 服务商与普通爬虫之间的协议，主要的作用便是为爬虫提供一个爬取的规范和限制。如果请求中出现了该 url 地址，我们即可将该请求标记为爬虫请求。除了 robots.txt, 我们在使用后文中出现的 taint link 技术时，也可能出现会触发一些异常 url 地址的访问，我们可以将这些标记为爬虫。

- **Http referer**: 该 http 请求字段标识着当前请求的访问来源，一般情况下，该 referer 字段指向的应该是当前网站中存在的某一网页。如果不满足该条件，则标记该请求的来源为爬虫。
- **Http 请求中的恶意攻击载荷**: 在真实的网络环境中，存在着大量漏洞扫描爬虫，这些爬虫发送的请求中往往带有恶意的攻击载荷，通过建立针对攻击载荷的数据库，我们很容易地过滤出这些爬虫。此处，我们参考的攻击载荷数据库来自于 360 天眼中的规则。详细的规则请参考附录 x
- **Http 请求头部字段的完备性**: 通常而言，大部分静态爬虫的主体（如 curl 以及一些基于 python-requests 库的简单爬虫）相对于浏览器而言功能负责度角度，这也意味在数据交换中，他们使用的功能字段更少，如下是来源 curl 的爬虫的访问请求，以及使用 chrome 的访问请求的完整 http 数据包的对比如



```

GET / HTTP/1.1
Host: 127.0.0.1:8889
User-Agent: curl/7.54.0
Accept: */*

GET / HTTP/1.1
Host: 127.0.0.1:8889
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36
Sec-Fetch-User: ?1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,zh-TW;q=0.7,pl;q=0.6,lb;q=0.5,da;q=0.4
  
```

图 5 curl 与 chrome 的访问请求对比

由此可以看出两者的 http 字段的复杂度相差很大。利用这点，我们可以对一些浏览器常用的字段检查，来过滤出一些简单的爬虫。在我们的模型中，我们使用的必要字段是：Host, Connection, User-Agent, Accept, Accept-Language, Accept-Encoding 没有使用这些字段的请求均会被标记为爬虫。

我们在数据集 a 和数据集 b 上使用单粒度模式检测规则的检测效果如下所示：

此处缺图.png

3.3 session 粒度模式检测

通过单粒度模式检测，我们可以过滤出互联网上大部分的爬虫。但是，爬虫与反爬虫本质上是一种博弈对抗，一旦爬虫采用一些字段修改和隐藏的方法，去修改我们检查的 `http` 特征，那么传统的单粒度模式检测就很发挥应有的作用，因此，我们还需要将我们检测的重点转向那些更难被修改却更加显著特征上，而这些特征主要富集在 `session` 粒度层次上。但是，大部分的关于爬虫识别的论文在 `session` 分类上都语焉不详，部分论文直接将来自于同 `ip` 的请求划分为同一个 `session`，而另一部分论文直接使用 `cookie` 来追踪浏览器的 `session`。这种处理显然是无法准确对恶意爬虫进行 `session` 分类的，恶意爬虫通常会使用分布式的 `ip` 地址，并对 `cookie` 部分的字段内容进行伪造，从而欺骗爬虫识别系统。在本节我们将介绍本研究在 `session` 分类上采用的策略和方案。

3.3.1 静态 session 分类

在 `session` 分类的处理上，本研究首先采用 `http` 字段静态特征作为 `session` 分类的依据。主要采用的静态特征有：`User-agent` 字段，`URL` 请求路径，其他 `header` 字段，`header` 字段键与其排列顺序。引入 `header` 字段排列的主要是基于这样的直觉：如果目标爬虫使用了 `http header` 字段变异的对抗行为，虽然会改变字段具体的值，但可能忽略对字段键的变异，给该特征赋予较高的权重，能够有效地在此类对抗行为下正确地分类 `session`。在计算两个 `http` 请求的相似度时，我们采用如下的计算函数：

$$score = \frac{W_{ua} * M_{ua} + W_{url} * M_{url} + W_{other} * M_{other} + W_{order} * M_{order}}{C_{field} + 2}$$

此处的 W 为权重相关的系数， M 表示相关字段是否匹配，匹配取值为 1，反之取值为 0，表示相关字段的数目，用以避免字段过多的请求出现评分过高的情况。此外，我们额外设置相应的阈值，一旦相似度超过该阈值，我们即可将两个 `http` 请求归类为同一个 `session`。我们在数据集 `a` 上测试我们的算法，并使用梯度下降的方法，求得这些参数的较优解。下图是我们使用该静态分类方法效果图（梯度下降图），我们最终采用的参数取值，其分类准确率可以达到 88

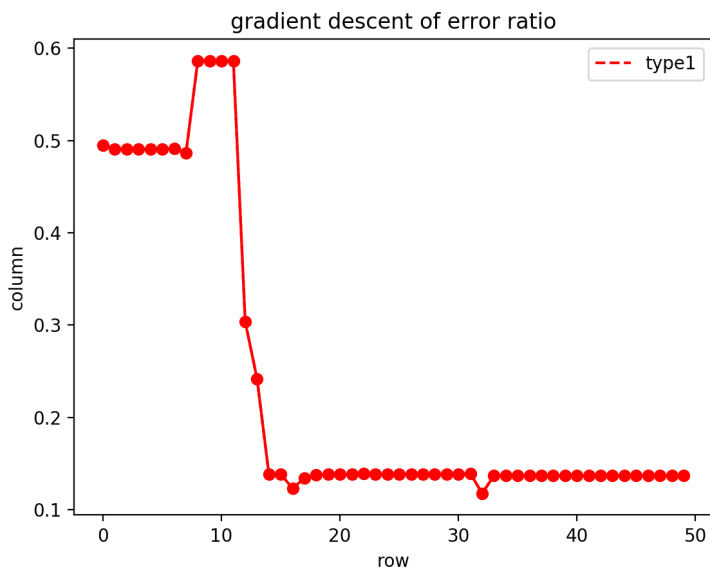


图 6 curl 与 chrome 的访问请求对比

静态 session 分类也是在已有的爬虫检测文献中最长使用的方法，但是方法也存在致命的局限性，在面对使用对抗策略的爬虫面前，该 session 分类方法无法正常地进行分类，下图是静态 session 方法在三种数据集上的分类效果：

表 2 静态 session 分类效果比较

数据集编号	数据特性	原始 session 数目	分类后 session 数目	session 分类准确率
a	正常用户 + 普通爬虫	100	130	88%
b	恶意漏洞扫描爬虫	100	291	35%
c	采用对抗策略的爬虫	100	472	26%

不难看出，在存在对抗样本的数据集 b 以及数据集 c 上，静态 session 分类的准确率都大打折扣。但是如果客户端 webdriver 开启了 javascript 执行的功能，那么通过引入动态 session 分类，能够有效地缓解上述的问题。

3.3.2 动态 session 分类

3.3.2.1 浏览器指纹生成算法

不同用户的浏览器具有不同的特征信息和丰富的数据，网站可以通过在用户访问时通过不同的 API 和技术手段获取浏览器特征信息构建独特的浏览器指纹。浏览器指纹技术能够在无 cookie 的情况下提供一种追踪用户的鲁棒性方案。令指纹生成算法为

$F(\square)$ 。当出现一个新的浏览器信息 x 时, 生成一个浏览器指纹 $F(x)$ 。该算法遵循离散概率密度函数 $P(f_n)$ 。其中, f_n 为某个特征信息的指纹生成结果; $n \in [0, 1, \dots, N]$, N 为特征信息的个数。对于某单个特征信息的指纹生成结果 f_n , 使用自信息量 I 表示该浏览器指纹所包含信息的比特数, 即:

$$I(F(x) = f_n) = -\log_2(P(f_n)) \quad (1)$$

当指纹由多个不同的特征信息组合而成时, 假设不同特征信息对应的指纹生成算法为 $F_s()$, $s \in [0, 1, \dots, S]$ (S 为指纹生成算法的个数), 则根据公式 (2) 单独计算每个特征信息的指纹生成结果 $f_{n,s}$ 的自信息量 $I_s(f_{n,s})$, 并根据公式 (3) 定义指纹组件的信息熵 $H_s(F_s)$ 。

$$I_s(f_{n,s}) = -\log_2(P(f_{n,s})) \quad (2)$$

$$H_s(F_s) = -\sum_{n=0}^N P(f_{s,n}) \log_2(P(f_{s,n})) \quad (3)$$

信息熵表征浏览器所有自信息量的期望值。对于两个相互独立的特征组件, 自信息量可直接线性相加。根据自信息量 I 可以确认指纹归属对象的身份。 I 包含的若干比特信息中的每一比特信息都能将该浏览器指纹可能的归属集合减半。特征信息进行组合生成指纹, 每一个特征信息具有若干比特的信息熵, 熵值越大, 则越能准确地区分不同的浏览器实体。因此要选取恰当且包含足够比特信息的特征信息集合, 通过该特征信息集合生成的指纹能够唯一确认指纹归属对象的身份。

3.3.2.2 Canvas 指纹

Mowery 和 Shacham 提出了通过 HTML 5 的 Canvas API 以及 WebGL 渲染文本得到像素差异, 作为一种具有高信息熵的浏览器指纹生成方式 [27]。canvas 元素是 html5 规范中新增的一类元素, 能够在屏幕上可编程地绘制图像, 并且被大部分主流的浏览器支持。绘制一个基本的 canvas 图形的方法相当简单: 需要浏览器提供给用户一个图形渲染的环境, 使用环境中的 API 来渲染你对 canvas 图像的操作和改变。在目前使用的 html5 规范中, 广泛使用的 2d 环境能提供诸如 fillRect, lineTo 以及 arc 这样的一些基本

绘图功能。也有一些复杂的功能可以支持贝塞尔曲线，颜色梯度的这样一些功能。

- **Canvas Text 渲染**：给定字体，字体颜色以及位置参数，2d context 能够在 canvas 绘制出任意的文本。
- **Canvas 像素抽取**：2D context 提供了一个 `getImageData` 方法，通过该方法能够获取一个给定区域范围内的图片对象，该对象是以图片中的每一个元素的 RGBA 值组成的。其次，canvas 对象本身提供 `toDataURL` 方法，当提供一个图片作为该方法的输入后，该方法能够将完整的图片内容以 base64 编码的形式返回。以上两个方法均严格遵循浏览器同源策略。
- **WebFont**：webFont 是定义在 CSS3 中的规范，允许用户按需加载远程字体，而不是只能依赖于已经安装在本地的字体。在使用这个特性时，需要加入 `@font-face` 这样一条规则，并使用 `src` 属性指定远程字体资源的 url 地址。为了使用 WebFont，需要开启 WebFont Loader 函数库。通过使用该函数库，WebFont 仅仅通过 JavaScript 就能够加载。
- **WebGL**：WebGL 提供了一个 javascript 的 API 用于在 canvas 上绘制图像，目前的主流浏览器都支持 WebGL 的硬件加速选项，可以使用图形硬件来渲染每一帧，目前 WebGL 也通过各自的 canvas 环境暴露出相应的函数接口，类似于 OpenGL API，可以使用 GLSL（OpenGL Shading Language）编程，在编译后，可以直接运行在硬件显卡上。

3.3.2.3 浏览器特定指纹

ECKERSLEY 根据浏览器的环境，通过收集一些网站能够获取的浏览器常用或者不常用的特性，用以生成浏览器的指纹 [26]。这些特性中有一部分可以通过简单静态的 HTTP 请求中推断出来，也可以经由 AJAX 接口收集。特性收集过程中，有一部分特性是简单明了的，但有一部分特性是来源于一些浏览器细节。有些浏览器禁用了 javascript，那么他会使用 video，plugins，fonts 和 supercookie 的默认值。因此可以通过这一细节来推断出浏览器是否禁用了 javascript。我们在研究中使用了如下的常见浏览器特性来产生浏览器指纹：

- user-agent
- 浏览器语言设置

- 浏览器 storage 设置 (SessionStorage, LocalStorage, IndexDb)
- cookie 支持
- 屏幕分辨率
- timezone
- 浏览器插件信息
- 系统字体
- doNotTrack 标识
- ie activex 支持
- canvas 指纹

以下是我们在不同操作系统，不同版本的不同浏览器上根据以上特征使用 hash 算法生成的 128 bit 指纹。

表 3 不同浏览器指纹比较

webdriver 名称	版本	操作系统	浏览器指纹
chromium	70.0.3509.0	ubuntu 16.04 x64	1a82cc57a97d7a931a82cc5720de9977
chromium	70.0.3514.0	ubuntu 16.04 x64	bae33d459368bee9bae33d45b8a2efa1
chromium	70.0.3519.0	ubuntu 16.04 x64	06a3007b36556ee506a3007bad572654
firefox(headless)	57	ubuntu 16.04 x64	576d92810cef4933576d9281c52a137b
firefox(headless)	63	ubuntu 16.04 x64	332ac9f169afb00b332ac9f18b81828d
firefox(headless)	68	ubuntu 16.04 x64	4cb3ae543ab898494cb3ae54fc189ce4
phantomjs	1.9.7	ubuntu 16.04 x64	2d15aa2ba520737e2d15aa2b4a84aa17
phantomjs	1.9.8	ubuntu 16.04 x64	4c83cbcb790af6734c83cbcb8e935880
phantomjs	2.1.1	ubuntu 16.04 x64	ba8506ab17e80837ba8506ab62082282
chrome	78.0.3904.97	macos 10.14.4	49c81cf7b7d05f9e49c81cf71fce9c5e
safari	12.1	macos 10.14.4	410160ac93cefc91410160ac72900fc9
firefox	70.0.1	macos 10.14.4	259295adde33bc05259295ad81fa3bed
chrome	78.0.3904.84	ios 12.2	cef500876945eee2cef50087fadd5d18

根据 Mowery 的论文 [12]，动态指纹的独特性和熵能够得到保证。因此，在并发流量不太高的情况下，将其作为 session 分类的重要依据，能够得到相当高的准确率。我们将动态指纹加入到我们的相似度计算函数，函数更新后为：

$$score = \frac{W_{ua} * M_{ua} + W_{url} * M_{url} + W_{other} * M_{other} + W_{order} * M_{order} + W_{dynamic} * M_{dynamic}}{C_{field} + 3}$$

因为我们使用的动态指纹分类技术需要 javascript 的支持，原先的数据集 a 和 b 并没有记录下相关的信息，因此我们只能使用数据集 c 来作为评估基于静态/动态指纹的 session 分类方法的准确性。在动态指纹生成的整个过程，我们首先在请求中捕获没有动态指纹生成、动态指纹过期、动态指纹伪造的请求，针对该请求发送相应的指纹生成 javascript 脚本脚本执行完后，会产生浏览器的特定指纹，并将该指纹返回到服务器（最好使用加密算法），服务器根据该指纹，分配并绑定 session。大致流程如下图所示：

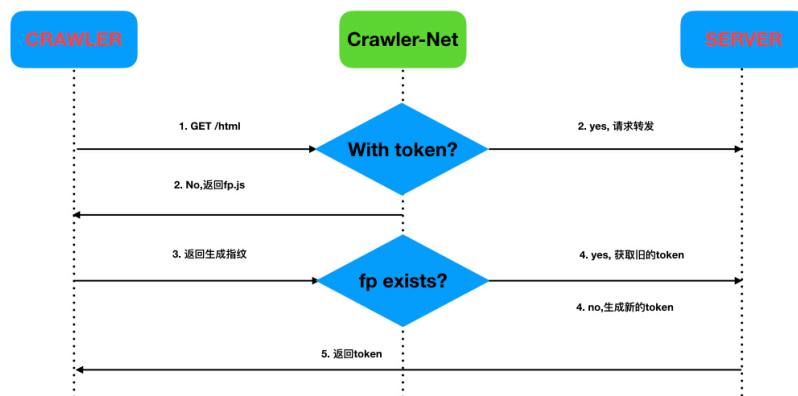


图 7 浏览器指纹生成流程

第四章 说明

Again, 这是北航论文 \LaTeX 模板 (CT \LaTeX -Based) B \LaTeX A \LaTeX H \LaTeX E \LaTeX S。

本 \LaTeX 模板为北航研究生学位论文模板, 适用于理工类博士、学术硕士和专业硕士。本 \LaTeX 模板参考自 2015 年 8 月版北航《研究生手册》, 具体要求请参见各自的《手册》, 最终成文格式需参考学院要求及打印方意见。本模板中大量内容和说明直接摘抄自《手册》(2015 年 8 月版), 基本覆盖了论文内容和格式方面的要求。

本模板已上传[GitHub](#), 该仓库中同时也包含了相应的 Word 模板。

4.1 宏包使用

请将以下文件与此 \LaTeX 文件放在同一目录中:

buaa.cls	▷ \LaTeX 宏模板文件
buaa_mac.cls	▷ \LaTeX 宏模板文件 (For Mac with Xe \LaTeX)
GBT7714-2005.bst	▷ 国标参考文献 Bib \TeX 样式文件 2005
GBT7714-2015.bst	▷ 国标参考文献 Bib \TeX 样式文件 2015
logo-buaa.eps	▷ 论文封皮北航字样
head-doctor.eps	▷ 论文封皮北博士学位论文标题
head-master.eps	▷ 论文封皮北学硕学位论文标题
head-professional.eps	▷ 论文封皮北专硕学位论文标题
tex/*.tex	▷ 本模板样例中的独立章节

通过 `\documentclass[<thesis>,<permission>,<printtype>,<ctexbookoptions>]{buaa}` 载入宏包:

thesis ▷ 论文类型 (thesis), 可选值:

- a) 学术硕士论文 (master) [缺省值]
- b) 专业硕士论文 (professional)
- c) 博士论文 (doctor)

permission ▷ 密级 (**permission**), 可选值:

- a) 公开 (**public**) [缺省值]
- b) 内部 (**privacy**)
- c) 秘密 (**secret=secret3**)
 - c.1) 秘密 3 年 (**secret3**)
 - c.2) 秘密 5 年 (**secret5**)
 - c.3) 秘密 10 年 (**secret10**)
 - c.4) 秘密永久 (**secret***)
- d) 机密 (**classified=classified5**)
 - d.1) 机密 3 年 (**classified3**)
 - d.2) 机密 5 年 (**classified5**)
 - d.3) 机密 10 年 (**classified10**)
 - d.4) 机密永久 (**classified***)
- e) 绝密 (**topsecret=topsecret10**)
 - e.1) 绝密 3 年 (**topsecret3**)
 - e.2) 绝密 5 年 (**topsecret5**)
 - e.3) 绝密 10 年 (**topsecret10**)
 - e.4) 绝密永久 (**topsecret***)

printtype ▷ 打印属性 (**printtype**), 可选值:

- a) 单面打印 (**oneside**) [缺省值]
- b) 双面打印 (**twoside**)

ctexbookoptions ▷ ctexbook 文档类支持的其他选项:

使用 ctexbookoptions 选项传递 ctexbook 文档类支持的其他选项。例如, 使用 fontset=founder 选项启用方正字体以避免生僻字乱码的问题¹。

模板已内嵌 LaTeX 工具包: ifthen, etoolbox, titletoc, remreset, remreset, geometry, fancyhdr, setspace, caption, float, graphicx, subfigure, epstopdf, booktabs, longtable, multirow, array, enumitem, algorithm2e, amsmath, amsthm, listings, pifont, color, soul, newtxtext, newtxmath。

¹需要系统安装方正字体。

模板已内嵌宏：`\highlight{text}`（黄色高亮）。

请统一使用 UTF-8 编码。

4.2 选项设置

`\refcolor` ▷ 开启/关闭引用编号颜色，包括参考文献，公式，图，表，算法等

on: 开启 [默认]

off: 关闭

`\beginright` ▷ 摘要和正文从右侧开始

on: 开启 [默认]

off: 关闭

`\emptypageword` ▷ 空白页留字

`\Listfigtab` ▷ 是否使用图标清单目录

on: 开启 [默认]

off: 关闭

4.3 章节撰写

本模板支持以下标题级别标题级别：

<code>\chapter{章}</code>	▷ 第一章
<code>\chapter*{无章号章}</code>	▷ 无章号章
<code>\chapter*{无章号有目录章}</code>	▷ 无章号有目录章
<code>\summary</code>	▷ 总结
<code>\appendix</code>	▷ 附录
<code>\achievement</code>	▷ 攻读学位期间取得的成果
<code>\acknowledgments</code>	▷ 致谢
<code>\biography</code>	▷ 作者简介
<code>\section{节}</code>	▷ 1.1 节
<code>\subsection{条}</code>	▷ 1.1.1 条
<code>\subsubsection{A}</code>	▷ 1.1.1.1 A
<code>\paragraph{a}</code>	▷ 1.1.1.1.1 a
<code>\subparagraph{a)}</code>	▷ 1.1.1.1.1.1 a)

4.4 注意事项

- ▷ 中文斜体将转换为楷体；
- ▷ buaa.cls 采用包 newtxtext 和 newtxmath，**中文粗体**在 Windows 下转换为黑体（有可能是因为 newtx 包没安装好，By WeiQM），Linux 下正常（By QiaoJF）；
- ▷ buaa_mac.cls 采用包 times，**中文粗体**转换为黑体（By CaiBW）；
- ▷ `\label{<text>}` 中不能使用中文；
- ▷ 浮动体与正文之间的距离是弹性的；
- ▷ 命令符与汉字之间请注意加空格以避免 undefined 错误（pdfLaTeX 下好像一般不存在这个问题，主要在 XeLaTeX 编译环境下发生）；

4.5 ToDo

- ▷ 数学环境的行间隔；
- ▷ 参考文献的行间隔；

4.6 意见及问题反馈

E-mail: weiqm@buaa.edu.cn

GitHub: <https://github.com/CheckBoxStudio/BUAAThesis/issues>

第五章 示例

5.1 参考文献引用

5.1.1 数字标注

<code>\cite{knuth86a}</code>	\Rightarrow	[?]
<code>\citet{knuth86a}</code>	\Rightarrow	?]
<code>\citet[chap.~2]{knuth86a}</code>	\Rightarrow	? , chap. 2]
<code>\citep{knuth86a}</code>	\Rightarrow	[?]
<code>\citep[chap.~2]{knuth86a}</code>	\Rightarrow	[? , chap. 2]
<code>\citep[see][]{knuth86a}</code>	\Rightarrow	[see ?]
<code>\citep[see][chap.~2]{knuth86a}</code>	\Rightarrow	[see ? , chap. 2]
<code>\citet*{knuth86a}</code>	\Rightarrow	?]
<code>\citep*{knuth86a}</code>	\Rightarrow	[?]
<code>\citet{knuth86a,tlc2}</code>	\Rightarrow	? ?]
<code>\citep{knuth86a,tlc2}</code>	\Rightarrow	[? ?]
<code>\cite{knuth86a, knuth84}</code>	\Rightarrow	[? ?]
<code>\upcite{knuth86a, knuth84}</code>	\Rightarrow	[? ?]
<code>\citet{knuth86a, knuth84}</code>	\Rightarrow	? ?]
<code>\citep{knuth86a, knuth84}</code>	\Rightarrow	[? ?]
<code>\cite{knuth86a, knuth84, tlc2}</code>	\Rightarrow	[? ? ?]

5.1.2 数字标注-上标形式

<code>\upcite{knuth86a}</code>	\Rightarrow	[?]
<code>\upcite{knuth86a, knuth84, tlc2}</code>	\Rightarrow	[? ? ?]

实现源码：`\newcommand{\upcite}[1]{\textsuperscript{\cite{#1}}}`。

5.1.3 著者-出版年制标

<code>\cite{knuth86a}</code>	\Rightarrow	?
<code>\citet{knuth86a}</code>	\Rightarrow	?
<code>\citet[chap.~2]{knuth86a}</code>	\Rightarrow	?, chap. 2
<code>\citep{knuth86a}</code>	\Rightarrow	(?)
<code>\citep[chap.~2]{knuth86a}</code>	\Rightarrow	(?, chap. 2)
<code>\citep[see][]{knuth86a}</code>	\Rightarrow	(see ?)
<code>\citep[see][chap.~2]{knuth86a}</code>	\Rightarrow	(see ?, chap. 2)
<code>\citet*{knuth86a}</code>	\Rightarrow	?
<code>\citep*{knuth86a}</code>	\Rightarrow	(?)
<code>\citet{knuth86a,tlc2}</code>	\Rightarrow	??
<code>\citep{knuth86a,tlc2}</code>	\Rightarrow	(??)
<code>\cite{knuth86a,knuth84}</code>	\Rightarrow	??
<code>\citet{knuth86a,knuth84}</code>	\Rightarrow	??
<code>\citep{knuth86a,knuth84}</code>	\Rightarrow	(??)

5.1.4 其他形式的标注

<code>\citealt{tlc2}</code>	\Rightarrow	?
<code>\citealt*{tlc2}</code>	\Rightarrow	?
<code>\citealp{tlc2}</code>	\Rightarrow	?
<code>\citealp*{tlc2}</code>	\Rightarrow	?
<code>\citealp{tlc2,knuth86a}</code>	\Rightarrow	??
<code>\citealp[pg.~32]{tlc2}</code>	\Rightarrow	?, pg. 32
<code>\citenum{tlc2}</code>	\Rightarrow	?
<code>\citertext{priv.\ comm.}</code>	\Rightarrow	[priv. comm.]
<code>\citeauthor{tlc2}</code>	\Rightarrow	?
<code>\citeauthor*{tlc2}</code>	\Rightarrow	?
<code>\citeyear{tlc2}</code>	\Rightarrow	?
<code>\citeyearpar{tlc2}</code>	\Rightarrow	[?]

5.2 浮动体

5.3 算法环境

模板中使用 `algorithm2e` 宏包实现算法环境。关于该宏包的具体用法请阅读宏包的官方文档。

—————↓—————Space Check—————↓—————

Data: this text

Result: how to write algorithm with $\text{\LaTeX}2\text{e}$
initialization;

while *not at end of this document* **do**

 read current;

if *understand* **then**

 go to next section;

 current section becomes this one;

else

 go back to the beginning of current section;

end

end

算法 1: A How to (plain).

—————↑—————Space Check—————↑—————

算法 2: A How to (ruled).

Data: this text

Result: how to write algorithm with $\text{\LaTeX}2\text{e}$
initialization;

while *not at end of this document* **do**

 read current;

if *understand* **then**

 go to next section;

 current section becomes this one;

else

 go back to the beginning of current section;

end

end

5.3.1 三线表

推荐使用三线表的方式，如表 5。

—————|—————Space Check—————|—————
—————|—————Space Check—————|—————

Data: this text
Result: how to write algorithm with L^AT_EX2e
initialization;
while *not at end of this document* **do**
 read current;
 if *understand* **then**
 go to next section;
 current section becomes this one;
 else
 go back to the beginning of current section;
 end
end

算法 3: A How to (boxed).

算法 4: A How to (boxruled).

Data: this text
Result: how to write algorithm with L^AT_EX2e
initialization;
while *not at end of this document* **do**
 read current;
 if *understand* **then**
 go to next section;
 current section becomes this one;
 else
 go back to the beginning of current section;
 end
end

表 4 表的标题

操作系统	TeX 发行版
所有	TeX Live
macOS	MacTeX
Windows	MikTeX

表 5 让我们看看一个长标题长什么样。还不够长? 那我再多写一点。还是不够长? 那我再多写一点。OK, 就是长这样的!

操作系统	TeX 发行版
所有	TeX Live
macOS	MacTeX
Windows	MikTeX

我们在这儿插入一行字;

我们在这儿再插入一行字;

我们在这儿插入一行字;

我们在这儿插入一行字；

我们在这儿插入一行字；

我们在这儿再插入一行字；

5.4 长表格

超过一页的表格要使用专门的 longtable 环境（表 6）。

—————↓—————**Space Check**—————↓—————

表 6 长表格演示

[illegible]

续下页

[illegible]

5.5 插图

北京航空航天大学

我们在这儿插入一行字；
我们在这儿再插入一行字；
我们在这儿插入一行字；

我们在这儿再插入一行字；

我们在这儿插入一行字；

我们在这儿再插入一行字；

我们在这儿插入一行字；

我们在这儿再插入一行字；

5.6 数学环境

5.6.1 数学符号

模板定义了一些正体（upright）的数学符号：

符号	命令
常数 e	<code>\eu</code>
复数单位 i	<code>\iu</code>
微分符号 d	<code>\diff</code>
$\arg \max$	<code>\argmax</code>
$\arg \min$	<code>\argmin</code>

更多的例子：

$$e^{i\pi} + 1 = 0 \quad (5.1)$$

$$\frac{d^2 u}{dt^2} = \int f(x) \, dx \quad (5.2)$$

$$\arg \min_x f(x) \quad (5.3)$$

5.6.2 定理、引理和证明

定义 5.1. If the integral of function f is measurable and non-negative, we define its (extended) **Lebesgue integral** by

$$\int f = \sup_g \int g, \quad (5.4)$$

where the supremum is taken over all measurable functions g such that $0 \leq g \leq f$, and where g is bounded and supported on a set of finite measure.

例 5.1. Simple examples of functions on \mathbf{R}^d that are integrable (or non-integrable) are given by

$$f_a(x) = \begin{cases} |x|^{-a} & \text{if } |x| \leq 1, \\ 0 & \text{if } |x| > 1. \end{cases} \quad (5.5)$$

$$F_a(x) = \frac{1}{1 + |x|^a}, \quad \text{all } x \in \mathbf{R}^d. \quad (5.6)$$

Then f_a is integrable exactly when $a < d$, while F_a is integrable exactly when $a > d$.

引理 5.1 (Fatou). Suppose $\{f_n\}$ is a sequence of measurable functions with $f_n \geq 0$. If $\lim_{n \rightarrow \infty} f_n(x) = f(x)$ for a.e. x , then

$$\int f \leq \liminf_{n \rightarrow \infty} \int f_n. \quad (5.7)$$

注. We do not exclude the cases $\int f = \infty$, or $\liminf_{n \rightarrow \infty} \int f_n = \infty$.

推论 5.2. Suppose f is a non-negative measurable function, and $\{f_n\}$ a sequence of non-negative measurable functions with $f_n(x) \leq f(x)$ and $f_n(x) \rightarrow f(x)$ for almost every x . Then

$$\lim_{n \rightarrow \infty} \int f_n = \int f. \quad (5.8)$$

命题 5.3. Suppose f is integrable on \mathbf{R}^d . Then for every $\epsilon > 0$:

1. There exists a set of finite measure B (a ball, for example) such that

$$\int_{B^c} |f| < \epsilon. \quad (5.9)$$

2. There is a $\delta > 0$ such that

$$\int_E |f| < \epsilon \quad \text{whenever } m(E) < \delta. \quad (5.10)$$

定理 5.4. Suppose $\{f_n\}$ is a sequence of measurable functions such that $f_n(x) \rightarrow f(x)$

a.e. x , as n tends to infinity. If $|f_n(x)| \leq g(x)$, where g is integrable, then

$$\int |f_n - f| \rightarrow 0 \quad \text{as } n \rightarrow \infty, \quad (5.11)$$

and consequently

$$\int f_n \rightarrow \int f \quad \text{as } n \rightarrow \infty. \quad (5.12)$$

证明. Trivial. □

5.6.3 自定义

Axiom of choice. Suppose E is a set and E_α is a collection of non-empty subsets of E . Then there is a function $\alpha \mapsto x_\alpha$ (a “choice function”) such that

$$x_\alpha \in E_\alpha, \quad \text{for all } \alpha. \quad (5.13)$$

Observation 5.1. Suppose a partially ordered set P has the property that every chain has an upper bound in P . Then the set P contains at least one maximal element.

A concise proof. Obvious. □

Observation 5.2. Suppose a partially ordered set P has the property that every chain has an upper bound in P . Then the set P contains at least one maximal element.

A concise proof. Obvious. □

我们在这儿插入一行字；

我们在这儿再插入一行字；

我们在这儿插入一行字；

我们在这儿再插入一行字；

我们在这儿插入一行字；

我们在这儿再插入一行字；

我们在这儿插入一行字；

我们在这儿再插入一行字；

我们在这儿插入一行字；
我们在这儿再插入一行字；
我们在这儿插入一行字；
我们在这儿再插入一行字；
我们在这儿插入一行字；
我们在这儿再插入一行字；
我们在这儿插入一行字；
我们在这儿再插入一行字；
我们在这儿插入一行字；
我们在这儿再插入一行字；

结 论

学位论文的结论单独作为一章，但不加章号。如果不可能导出应有的结论，也可以没有结论而进行必要的讨论。

* 嗯，这就是你的论文了 *

附 录

下列内容可以作为附录：

- 1) 为了整篇论文材料的完整，但编入正文又有损于编排的条理和逻辑性，这一材料包括比正文更为详尽的信息、研究方法和技术更深入的叙述，建议可以阅读的参考文献题录，对了解正文内容有用的补充信息等；
- 2) 由于篇幅过大或取材于复制品而不便于编入正文的材料；
- 3) 不便于编入正文的罕见的珍贵或需要特别保密的技术细节和详细方案（这中情况可单列成册）；
- 4) 对一般读者并非必要阅读，但对专业同行有参考价值的资料；
- 5) 某些重要的原始数据、过长的数学推导、计算程序、框图、结构图、注释、统计表、计算机打印输出文件等。

* 嗯，自由发挥吧 *

攻读硕士学位期间取得的学术成果

对于博士学位论文，本条目名称用“攻读博士学位期间取得的研究成果”，一般包括：

攻读博士学位期间取得的学术成果：攻读博士学位期间取得的学术成果：列出攻读博士期间发表（含录用）的与学位论文相关的学位论文、发表专利、著作、获奖项目等，书写格式与参考文献格式相同；

攻读博士期间参与的主要科研项目：列出攻读博士学位期间参与的与学位论文相关的主要科研项目，包括项目名称，项目来源，研制时间，本人承担的主要工作。

对于硕士学位论文，本条目名称用“攻读硕士学位期间取得的学术成果”，只列出攻读硕士学位期间发表（含录用）的与学位论文相关的学位论文、发表专利、著作、获奖项目等，书写格式与参考文献格式相同。

* 嗯，研究生不列科研项目 *

致 谢

致谢中主要感谢指导教师和在学术方面对论文的完成有直接贡献及重要帮助的团体和人士，以及感谢给予转载和引用权的资料、图片、文献、研究思想和设想的所有者。致谢中还可以感谢提供研究经费及实验装置的基金会或企业等单位 and 人士。致谢辞应谦虚诚恳，实事求是，切记浮夸与庸俗之词。

* 嗯，感谢完所有人之后，也请记得感谢一下自己 *

作者简介

博士学位论文应该提供作者简介，主要包括：姓名、性别、出生年月日、民族、出生的；简要学历、工作经历（职务）；以及攻读博士学位期间获得的其他奖项（除攻读学位期间取得的研究成果之外）。

* 嗯，“硕士学位论文无此项”，《手册》上是这么说的 *

This is B_UA_TH_ES, Happy TeXing! — from WeiQM.