计算机研究与发展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202110681
60(4): 839−859, 2023

Web 追踪技术综述

王晓茜[1,2] 刘奇旭[1,2] 刘潮歌[1,2] 张方娇[1,2] 刘心宇[1,2] 崔 翔[3]
[1](中国科学院信息工程研究所 北京 100093)
[2](中国科学院大学网络空间安全学院 北京 100049)
[3](广州大学网络空间先进技术研究院 广州 510006)
(wangxiaoxi@iie.ac.cn)

Survey of Web Tracking

Wang Xiaoxi[1,2], Liu Qixu[1,2], Liu Chaoge[1,2], Zhang Fangjiao[1,2], Liu Xinyu[1,2], and Cui Xiang[3]
[1](*Institute of Information Engineering*, *Chinese Academy of Sciences*, *Beijing* 100093)
[2](*School of Cyber Security*, *University of Chinese Academy of Sciences*, *Beijing* 100049)
[3](*Cyberspace Institute of Advanced Technology*, *Guangzhou University*, *Guangzhou* 510006)

Abstract Web tracking has become a research hotspot under the background of information age, and it is an important means for user identification and behavior analysis. By following up the research achievements in this field, we analyze the current status of researches and development situation in the field of Web tracking from the aspects of tracking technology or defense technology. Firstly, we introduce the current researches according to the implementation of technology, after dividing the Web tracking into storage tracking technology and fingerprint tracking technology. Secondly, according to the tracking ability of different technologies, we divide Web tracking technology into three different levels: single-browser tracking, cross-browser tracking and cross-device tracking, then we analyze and discuss the features' acquisition technology and attribute characteristics of the features, expound the relationship among features, correlation technology and the scope of tracking technology. From the perspective of Web tracking defense technology, we describe the different technology implementation characteristics and defense countermeasures of defense of browser extension, browser-embedded defense, defense framework tools and other mechanisms, defense countermeasures or environment. Finally, we summarize the current research situation in the field of Web tracking, targetedly analyze the advantages and disadvantages of Web tracking technology and the Web defense technology, and point out the current problems and possible development directions in this field.

Key words browser fingerprinting; Web tracking technology; Web tracking defense technology; fingerprint correlation algorithm; feature acquisition technology

摘 要 Web 追踪技术已经成为信息化时代背景下的研究热点,是对用户进行身份标识和行为分析的重要手段.通过跟进该领域的研究成果,从追踪技术和防御技术 2 方面分析 Web 追踪领域的研究与发展现状.首先按照技术的实现方式将 Web 追踪分为了存储型追踪技术和指纹型追踪技术,分析了当前研究追

收稿日期: 2021-06-11; **修回日期**: 2022-06-23
基金项目: 中国科学院青年创新促进会(2019163); 国家自然科学基金项目(61902396); 中国科学院战略性先导科技专项项目(XDC02040100); 中国科学院网络测评技术重点实验室和网络安全防护北京市重点实验室项目
This work was supported by the Youth Innovation Promotion Association of Chinese Academy of Sciences (2019163), the National Natural Science Foundation of China (6190396), the Strategic Priority Research Program of Chinese Academy of Sciences (XDC02040100), and the Project of the CAS Key Laboratory of Network Assessment Technology and Beijing Key Laboratory of Network Security and Protection Technology.
通信作者: 刘奇旭(liuqixu@iie.ac.cn)

踪现状.其次按照追踪范围将Web追踪技术分为单浏览器追踪、跨浏览器追踪、跨设备追踪3个不同的层次,分析和讨论特征的获取技术和属性特点,论述特征、关联技术、追踪范围的关系;同时从Web追踪防御技术的形态角度,描述扩展防御、浏览器内嵌防御、防御框架工具和机制、防御对策或环境等不同技术的实现特点和抵御追踪的措施.最后总结现有研究概况,针对性分析Web追踪技术和Web防御技术的优劣势,指出当下面临的问题及可能的发展方向.

关键词 浏览器指纹;Web追踪技术;Web追踪防御技术;指纹关联算法;特征获取技术

中图法分类号 TP391

Web新技术不断地更新迭代,以及信息化时代下用户信息的爆炸式增长,直接促进了Web追踪技术的高速发展.Web追踪技术通过为用户生成唯一的身份标识,实现对用户以及行为活动的追踪.其最初目的是服务不同的网站应用追踪用户,分析挖掘用户行为特点,实现个性化推荐.但追踪技术本身存在一定的安全问题,且易被恶意利用获取用户敏感信息.随着用户隐私敏感度的提升和信息保护意识的增强,以保护用户隐私为目标的追踪防御技术也在不断地升级和快速发展.

自2010年浏览器指纹[1]技术的提出,Web追踪技术的研究趋势从Cookie等类型的在用户客户端(即浏览器)存储身份标识的有状态的追踪,进入了基于浏览器获取特征信息的无状态[2]的指纹追踪时代.与传统的Cookie类追踪技术相比,指纹追踪技术摒弃了在客户端存储信息的形式,而是获取了较多具有用户个人特点的、多元化的特征信息,利用一种或多种不同类型的特征信息,生成唯一的指纹,在用户无感知的情况下进行隐蔽追踪.现阶段Web追踪领域的学术研究较多专注于某项特定的追踪或防御技术,因此,本综述针对Web追踪领域的技术发展和研究进行了总结和分析,并讨论了现有技术发展的优劣势,对该领域的下一步研究方向进行了展望.

本文研究了2010—2021年期间Web追踪领域的技术发展,按照实现机制将Web追踪技术分为了存储型追踪技术和指纹型追踪技术,从追踪技术的起源、技术实现、发展现状等多个角度进行介绍分析.着重分析Web追踪及防御技术的研究及实现方案,将Web追踪技术按照追踪范围划分为3个层次,描述了不同追踪范围所使用特征的特点和获取方式.从防御技术的不同形态进行分类介绍,对防御的技术实现进行深入分类讨论.着眼于Web追踪和防御技术2个角度,分析其技术上的优劣性以及当前面临的挑战和问题,指出未来可能的研究方向.

本文的主要贡献有4个方面:

1)介绍了Web追踪的起源和分类,从存储型追踪和指纹型追踪2方面介绍Web追踪技术的发展和特点.简要描述了现阶段Web追踪技术的实现和发展历程.

2)针对近些年Web追踪技术的发展进行深入调研,按照追踪范围将Web技术追踪分为了单浏览器、跨浏览器、跨设备3个不同层次.基于追踪范围分别介绍指纹中的特征获取技术,描述指纹关联算法并对现有的指纹追踪进行讨论.

3)深入研究Web追踪防御技术,按照形态可将防御技术分为扩展防御,浏览器内嵌防御,防御框架、工具或机制、构建防御对策或环境.分别介绍不同形态在防御技术上的实现方式,通过对比防御能力以及抵御追踪的手段,分析了Web追踪防御的现状.

4)基于Web追踪技术和防御技术2个角度分析了技术发展的优劣势,以及面临的挑战和发展方向.在追踪技术方面进行了唯一性和持久性讨论;在防御技术方面针对防御形态和技术进行对比,分析防御技术的优缺点,并指出现有技术无法完全防止指纹追踪.

1 Web追踪概述

早期Web追踪技术的出现是为了保持用户与服务器之间的会话状态,在用户浏览器客户端中存储一定的用户隐私信息,在交互式Web中对用户进行识别和区分.例如Cookie, FlashCookie, EverCookie, HTML5中新型的存储型API等均使用在客户端存储身份标识的方式进行追踪,我们将这种技术统称为存储型追踪技术.2010年电子前沿基金会(electronic frontier foundation, EFF)的专家Eckersley[1]提出的一项划时代的Web追踪技术——浏览器指纹,其基于用户浏览器获取设备的软硬件特征,并结合现有的存储型追踪机制,在服务器端对用户进行唯一性识别和追踪.随后,Web追踪领域掀起了使用各种不同

类型特征信息作为指纹来追踪用户的研究热潮，我们将这类追踪技术统称为指纹型追踪技术. 依据 Web 追踪的发展历史以及追踪技术手段的不同，本节将 Web 追踪技术分为 2 类进行概述：存储型追踪技术和指纹型追踪技术[3].

1.1 存储型追踪技术

Cookie 诞生自 1993 年，是使用较为广泛且最具代表性的存储型追踪技术，其最有效的功能是在网站中保持用户状态，提供用户个性化服务. 但直至 2000 年网络隐私权的提出，Cookie 才开始引起社会的关注，之后诞生了一系列以存储关键信息为手段的追踪技术，包括 FlashCookie, EverCookie, 随后 HTML5 新技术也开放了更多的权限给开发者[4]，例如增添了可直接在浏览器中进行内容存储和数据库存储的 API，如表 1 所示. 这些存储机制也为追踪提供了更加便利的条件.

Table 1 Analysis of Storage Tracking Technology Capabilities

表 1 存储型追踪技术能力分析

存储型追踪	使用范围（常规浏览器）	是否跨域共享	多位置置存储	是否多浏览器共享	存储时长	其他
Cookie	全部	是（可使用前后端技术实现）	否	否	设置过期时间	安全性较低
Flash Cookie	全部	是	是	是	默认无过期时间	已弃用
UserData	Windows + IE5	否	否	否	设置过期时间	已弃用
EverCookie	全部	是	是	是（使用 Flash Cookie 机制）	可通过各种存储机制重建	已弃用
SessionStorage	除 IE8 以下版本之外浏览器	否	否	否	浏览器会话时效内	安全性较低
LocalStorage	除 IE8 以下版本之外浏览器	否	否	否	非主动删除不过期	安全性较低
Web SQL	部分	否	否	否	无限制	已弃用
IndexedDB	全部	否	否	否	无限制	安全性较低

1.1.1 Cookie 类存储技术

HTTP Cookie 通常存储在 Web 客户端特定目录下，通过在与 Web 服务器的交互请求中携带 Cookie 值来辨别身份. 在设置并存储 Cookie 后，当用户再次登录网站时，浏览器都会检查本地的 Cookie 值，在 Web 请求中携带 Cookie 值发送给 Web 服务器. 服务器返回用户对应的个性化信息和其他权限内资源.

Flash Cookie[5] 同样是记录用户浏览网页时保留的信息，是由 FlashPlayer 控制的客户端共享存储技术 Share Object, 它永久驻留在用户计算机上，使不同的浏览器共用同一个 Flash Cookie, 实现实时数据共享. Flash Cookie 使用多个存储位置，较为隐蔽且难以清除. Flash Player 在 2020 年已经被正式淘汰.

UserData 是微软为 IE 浏览器专门在系统中开辟的一块存储空间，可用于对数据进行持久化存储，除非手动删除，否则即使 IE 浏览器被关闭或者刷新，数据也能够重新被载入而不会丢失，并且永远不会失效. 微软已经放弃了对 IE 浏览器的支持，将内置浏览器切换为 Chromium 内核的 Microsoft Edge 浏览器.

EverCookie[6] 又称为“僵尸 Cookie”，是一种更甚于 FlashCookie 的浏览器持久化追踪机制，它使用多达 10 种以上的存储机制将 Cookie 持久化存储在不同的位置，当在某个位置的 Cookie 被清除后，还可以通过其他不同位置的 Cookie 进行恢复. EverCookie 所使用的技术不仅包含 HTTP Cookie, UserData, Flash Cookie, 还囊括了 HTTP 头中的 ETags 和 HTML5 中的 SessionStorage, LocalStorage, IndexDB, Java 硬盘写入漏洞等多个存储机制. Acar 等人[7] 早在 2014 年针对 EverCookie 的研究中，就发现了 EverCookie 启用 IndexDB 技术进行存储，但 EverCookie 的野心不止于此，未来可能还会加入超文本传输协议（Hypertext Transfer Protocol, HTTP）身份验证缓存、TLS 会话恢复（TLS session resumption）标识符、HTTP 公钥固定（Http public key pinning, HPKP）认证等存储形式.

1.1.2 HTML5 新型存储型 API

Web Storage[8] 是 HTML5 提出的存储型 API, 包括会话存储 SessionStorage 和本地存储 LocalStorage, 可在用户客户端保存大量的数据，数据仅存储在客户端且不与服务器产生交互，每个域都有独立的存储空间. 其中 SessionStorage 仅在当前会话下有效，关闭了浏览器窗口就会被销毁；而 LocalStorage 的生命周期是永久的，关闭页面或浏览器之后，数据也不会被清除，除非主动删除数据，否则数据永远不会消失，且 2 种存储机制均遵循同源机制.

Web 数据库存储包括 Web SQL 和 IndexedDB[9]2 种类型，提供了在客户端直接进行数据存储的能力，理论上没有存储大小限制. Web SQL 是前端的独立的功能模块，具有与实际数据库基本一致的增删改查等操作. IndexedDB 是为了弥补 LocalStorage 存储缺陷诞生的，是浏览器提供的非关系型本地数据库. 可储存大量数据，提供查找接口和建立索引，但不支持关系型查询语句. IndexedDB 使用键值对存储数据，可进行异步操作，支持事务操作，受同源限制，存储空间大，支持二进制存储.

1.2 指纹型追踪技术

指纹型追踪技术，是一种通过浏览器获取用户设备暴露给网站的可见特征信息，生成唯一标识用户的指纹，来跟踪用户的方法. 当用户访问并加载 Web 网页时，网站利用浏览器的能力获取设备、软件相关配置、用户个性化信息以及 Cookie 等存储型追踪标识等，将这些特征 Hash 为一个唯一值传递到服务器作为用户的唯一标识. 其依托于用户浏览器对 Web 网站页面的渲染和执行能力，以及设备对浏览器权限和能力的支持.

1.2.1 特征获取与指纹生成

JavaScript 和 HTML5[10] 的 API 为追踪者提供了无需经过用户同意可直接获取用户设备和软件信息的能力. 例如可通过 Navigator 对象获取浏览器的用户代理、首选语言、是否设置不追踪(do not track, DNT)等，Screen 对象和 Date 对象描述了浏览器所处设备的信息，包括屏幕、时区等，如表 2 所示.

Table 2 API of JavaScript Object and the Corresponding Obtained Information

表 2 JavaScript 对象的 API 及对应获取的信息

JavaScript 对象	API	获取的信息
Navigator（宿主对象）	navigator.platform	系统平台
	navigator.userAgent	用户代理
	navigator.language	浏览器首选语言
	navigator.cpuClass	浏览器 CPU 等级
	navigator.plugins	插件列表
	navigator.doNotTrack	是否设置不追踪
Screen（宿主对象）	screen.width/screen.height	屏幕分辨率
	screen.availWidth/screen.availHeight	屏幕可用分辨率
	colorDepth	色彩深度
Date（内置对象）	*getTimezoneOffset*( )	时区

Eckersley 依托于前端语言的能力，构建了获取 8 项用户特征信息的指纹脚本，计算得出特征信息值至少包含 18.1b 的信息熵，使用 Murmurhash 算法[11] 将 8 种不同特征的信息 Hash 作为唯一的值，指纹出现重复的概率只有 1/286777. 在指纹生成后使用 difflib 库对指纹进行简单的关联. 通过大规模的数据收集和实验论证，结果显示指纹识别率可达到 83.6%，在开启 Java 和 Flash 的情况下，指纹识别率可高达 94.2%.

1.2.2 技术发展趋势

唯一性和稳定性是衡量追踪技术优劣的 2 大指标，指纹型追踪技术虽然弥补了一部分存储类追踪技术的缺点，减少了用户信息被窃取和欺骗的可能，但其本身存在较多不稳定因素：使用过程中的常规软件升级安装、添加插件扩展、更改分辨率、网络配置等均会影响指纹的唯一性和稳定性.

业界内也面向指纹追踪技术本身的问题展开了研究，技术的发展趋向于在指纹中使用不太稳定但含有较高信息量和较多用户特点的信息，如历史记录、字体、浏览器扩展等，以及较为稳定但唯一性较低的特征，如操作系统特征、硬件特征. 指纹型追踪技术结合现有的 Cookie 追踪、HTML5 的存储类 API 等存储型技术特性，来提高指纹特征包含的信息量以及指纹所具备的独特性和稳定性. 如表 3 所示，按照特征属性将研究中获取的特征分为了 5 大类：浏览器特征、操作系统特征、网络特征、硬件特征、存储型追踪标识.

浏览器指纹技术的发展同样也带来了新的用户隐私问题，因此 Web 追踪防御技术迎来了技术的发展热潮. 研究学者们从使用安装扩展、防御框架、工具，以及浏览器内嵌防御等不同的形态和方式着手，针对性缓解一种或多种的指纹特征获取技术，减少被持续追踪的可能性，防止用户隐私被泄露. Web 追踪技术和防御技术在不断对抗中快速发展，但技术的成熟还面临着诸多的问题和挑战.

2 Web 追踪技术

指纹型追踪技术已经将存储型追踪技术以及常用于溯源的 IP 地址、网络等信息纳入了指纹的特征范畴，且指纹型的特征信息获取技术已经从单纯的 API 调用，延伸到了使用枚举、测量、漏洞利用等手段获取用户设备上的各种硬件和软件信息[12]，以及使用指纹关联算法延长追踪持久性. 本节着眼于 Web 追踪技术的追踪范围，首先将 Web 追踪技术分为单浏览器追踪、跨浏览器追踪，跨设备追踪 3 个不同层次，针对较为复杂的特征及获取方式(不可通过 API

Table 3 Classification and Introduction of Fingerprint Tracking Technology Features

表 3 指纹型追踪技术特征分类及介绍

特征分类	可利用特征	变化因素	获取方式
浏览器特征	用户代理（user agent，UA）	随浏览器升级而变化，短期内稳定	API
	语言	与用户设置相关，相对稳定	API
	字体	与系统安装字体相关，不稳定	枚举探测
	历史记录、浏览器缓存	随用户使用习惯变化，可清除，不稳定	枚举探测
	插件	随系统安装软件变化，不稳定	API
	扩展	随用户喜好和使用率变化，不稳定	枚举探测
操作系统特征	操作系统类型	与系统安装相关，较为稳定	API
	时区	与用户设置相关，相对稳定	API
	主机缓存	与用户行为相关，短期内相对稳定	枚举、间接利用
网络特征	公网 IP	与网络环境相关，不稳定	HTTP 头获取
	内网 IP、内网主机信息及开放端口	与内网环境相关，不稳定	漏洞利用、枚举探测
	TLS 会话追踪	与网站和浏览器设置相关，相对稳定	攻击利用
硬件特征	硬件平台、分辨率、色彩深度	与硬件本身相关，短期内无变化，较为稳定	API
	CPU、GPU、音频、电池	与硬件属性和性能相关，较为稳定	API、测量获取
用户交互特征	是否设置“不追踪”、鼠标键盘记录	与用户操作相关，不稳定	算法分析
存储型追踪标识	Cookie、HTML5 存储型 API	与用户操作相关，可清除，相对稳定	API

调用获取的特征)进行介绍. 同时介绍了可有效弥补了指纹中追踪唯一性的不足的指纹的关联技术, 以及特征与追踪范围之间的关系.

2.1 单浏览器追踪

浏览器作为 Web 的入口, 也是信息被暴露的直接出口. 基于 Web 浏览器所包含的用户特点、浏览器引擎及其所依赖的硬件条件, 实现对用户浏览器的唯一标识, 即浏览器范围内的用户追踪.

2.1.1 含有较为丰富信息的特征

近些年的 Web 浏览器追踪的研究热点集中在利用含有较多用户个人特点, 且具有高信息熵的特征来产生较为稳定和持续的指纹, 例如扩展、字体、历史记录、浏览器缓存等. 这些特征可在一个浏览器内实现对用户的追踪.

1) 历史记录

Felten 等人[13]介绍了一种可以损害用户浏览器历史记录隐私的攻击, 使得恶意网站可以嗅探用户对其他站点的访问[14], 测量用户浏览器执行某些操作所需的时间来确定用户的浏览历史记录, 进而对用户身份去匿名化, 例如利用社交网站上可用的组成员身份信息唯一地标识此人, 或者显著减少候选人集[15].

利用层叠样式表(cascading style sheets, CSS)中:visited 样式表对页面中嵌入的“已访问”和“未访问”的地址链接(uniform resource locator, URL)颜色进行差异表示的漏洞[16], 能够对一定数量的 URL 地址进行枚举, 获取用户的访问列表. Janc 等人[17]基于链接样式显示区别, 使用了 3 种方式对历史记录进行嗅探, 包括直接利用链接访问样式差异、间接为已访问链接和未访问链接设置不同渲染图片、时序攻击. Olejnik 等人[18]通过大规模数据研究论证了 69% 用户的浏览历史是唯一的, 且奇异值大于 18b, 具有较强的稳定性. 随后 Yan[19]在圣地亚哥的安全会议 Toorcon 上示范了利用 HTTP 严格传输安全协议(Http strict transport security, HSTS)和内容安全策略(content-security-policy, CSP)2 种安全策略探测历史记录的漏洞工具, 并将其命名为 Sniffly. Smith 等人[20]提出了 4 种新的历史记录嗅探攻击: CSS Paint API、CSS 3D transforms、fill-coloring of SVGs 构造链接点击攻击、字节码缓存攻击, 并针对主要浏览器和具有安全防护性的浏览器进行了攻击测试, 发现依然可以取得很高的信息泄露率. Huang 等人[21]在 Smith 的方法技术上引入了多个辅助环节的自适应方法, 并采用动态参数搜索算法对 Smith 方法进行了改进, 实验结果表明, 在最流行的操作系统和平台上的 5 种主要浏览器上, 该自适应方法能够达到近 100% 的精度.

2)字体

Eckersley 使用 Adobe Flash 和 Java 虚拟机(Java

virtual machine, JVM)来探测系统中安装的字体的完整列表，并且在可检测字体的浏览器中，排序后字体的熵值为 16.0. Boda 等人[22]从将字体作为指纹的一个输入项的角度，分析了 JavaScript 的字体检测，发现字体列表可以为识别提供坚实的基础，并且论证了 IP 地址、特定字体集、屏幕分辨率、时区足够唯一的识别 5 种流行的浏览器. Fifield 等人[23]指出浏览器中的字体渲染受到浏览器版本、安装字体以及抗锯齿设置等多方面的因素影响而导致，作者提出了一种基于测量字体字形的屏幕尺寸来对字体进行枚举，证实即使是相对较粗糙的测量字形边界框的工具也可以产生很强的指纹，且设备上的字体信息可用于识别已安装的应用程序以及操作系统的类型和版本[24].

3)扩展

用户安装浏览器扩展本意是为了更便捷地服务，或更好地防止用户隐私泄露，但扩展的可枚举性在无形之中为用户带来更多被泄露的隐私. 相比其他含有丰富信息量的特征，扩展能够为研究学者提供较为确定的结果，且便于进行大规模的数据研究.

依据对扩展检测方式的不同，可以将枚举探测分为 2 类：静态探测和动态探测. 静态探测通过枚举扩展 ID、Web 可访问资源(Web accessible resources, WAR)或通过 URL 泄露来探测扩展. 通过在用户浏览器中对扩展 ID 的既定路径进行探测[25]，可枚举浏览器中的扩展. 在 Firefox 使用随机化 ID 的方式来避免扩展被探测的情况下，依然可以通过检测扩展的 WAR 路径判定扩展是否存在[26]，识别 90% 的扩展[27]. Sanchez-Rola 等人[28]讨论了统一资源标志符(uniform resource identifier, URI)泄露技术，颠覆 Safari 中实现的 URI 随机化机制，影响了多达到 40% 的 Safari 扩展. 动态探测方式主要通过监控扩展在页面上做出的改变，判断扩展是否存在. XHOUND[29]通过量化监测扩展在页面文档对象模型(document object model, DOM)做出的修改进行探测和识别，这是第 1 个对浏览器扩展进行指纹识别的全自动系统. Karami 等人[30]基于已有检测 WAR 路径和页面 DOM 独特修改 2 种技术，结合传出 HTTP 请求和浏览器内部信息交换这 2 种新颖的扩展检测方式，构建了一个基于行为扩展的全自动创建和检测指纹的工具 Carnus，并且使用 Carnus 创建了迄今为止最大的扩展指纹集. Laperdrix 等人[31]利用扩展在 Web 页面注入 CSS 规则的操作，提前在 Web 界面中注入 CSS 规则，通过检测碰撞对扩展进行枚举，这种方式可以识别之前技术中未检测出的 1 074 个扩展. 浏览器扩展枚举技术研究进展汇总如表 4 所示.

Table 4 Researches on Technologies of Enumerating Browser Extension

表 4 浏览器扩展枚举技术研究

研究技术	扩展数据集	技术形式	功性能
XHOUND[29]	10 000 个最流行的 Google Chrome 扩展	对页面 DOM 进行的独特的修改进行检测	几秒钟内对数十个扩展进行指纹识别
Discovering Extensions via WARs[26]	43 429 个 Chrome 扩展	通过 Web 可访问资源，检测扩展是否存在	可检测前 1 000 个免费的 Chrome 扩展中的 50%以上和所有 Chrome 扩展的 28%
Extension breakdown[28]	718 个 Safari 扩展	URI 泄露技术	可识别 40% 以上的 Safari 扩展
Latex Gloves[27]	62 994 个 Chrome 扩展，8 646 个 Firefox 扩展	WAR 检测，以及检测扩展在网页中注入的代码	能够识别 90% 的内容注入型扩展
Carnus[30]	29 428 个可检测的 Chrome 扩展	4 种不同的检测技术	83.6%~87.92% 的指纹在最先进的对策下仍然有效
Fingerprinting in Style[31]	116 485 Chrome 扩展	注入 CSS 的枚举插件 Extension	可识别 4 446 个扩展，且有 1 074 个扩展（24%）未被之前的技术识别

4)API 属性枚举和检测

Nikiforakis 等人[32]证实了在不同浏览器家族、版本中，甚至在不同操作系统中相同的浏览器版本中，特殊对象的属性枚举顺序始终是不一致的，这种不一致性可以作为识别用户的依据来对用户进行追踪，基于此提出了一种基于浏览器 JavaScript 对象属性枚举的指纹追踪方法. Mulazzani 等人[33]通过分析浏览器是否符合 JavaScript 标准，检测其支持的功能，从 150 多种浏览器和操作系统组合收集的数据集中，计算出了需要运行的最小测试集，以唯一地标识每个组合.

即便 JavaScript 被禁用依然能够对用户进行指纹追踪[34]，利用 CSS 不同的 API 属性在不同渲染引擎中的展现形式的差异，使用字符串请求、媒体查询、字体规则检测等技术，收集浏览器的类型、版本、设备屏幕分辨率、字体等信息来生成指纹. Schwarz 等人[35]提出了 2 种新的侧信道攻击来检测指令集体系和使用的内存分配器，使用一种完全自动化的手段

发现由环境引起的浏览器引擎的细微差别. JavaScript 引擎的固有性能签名也能够在传统形式的系统标识(例如用户代理标头)被修改或隐藏的情况下，检测浏览器版本、操作系统、微体系架构等信息[36].

5)浏览器缓存信息利用

Solomos 等人[37]将网站头像(favorites icon, favicons)作为强大的跟踪向量，向量中的每一位对应用户在 1 个特定子域上的图标请求，将该信息生成 32b 跟踪标识符，唯一标识和追踪用户. Bansal 等人[38]使用了细粒度的超时和 HTML5 Web Workers 来对网站进行并行化缓存探测，并在缓存请求中使用超时以防止缓存污染提高攻击效率，证实了缓存定时攻击的适用性和有效性.

6)交互式特征

文献 [39−41] 监控和跟踪鼠标的移动和点击，捕获用户与 Web 应用程序交互时的鼠标操作，推断用户输入的 URL，评估 Web 应用程序的有用性和易用性[42]，推断用户注意力变化，对定量数据进行预测. Lipp 等人[43]介绍了通用按键定时攻击，能够使攻击在后台选项卡或最小化的浏览器窗口中运行，以此观察到用户个人识别码(personal identification number, PIN)或密码的确切击键时间，推断用户输入的 URL，区分分时共享计算机的不同用户.

2.1.2 较为稳定的高级追踪特征

近些年陆续出现的 Canvas 指纹、WebGL(Web graphics library)指纹、Audio 指纹是当前被广泛使用的且较为稳定的指纹追踪方式. 这些追踪技术依赖于设备的硬件性能，但会受浏览器的影响.

1)Canvas 指纹

2012 年 Mowery 等人[44]提出了 Canvas 指纹，揭露了不同硬件设备对字体和文本的图片场景渲染的差异性. Canvas 图片处理依赖于浏览器图像处理引擎、压缩等级和导出选项，因此在不同设备上的绘制的图案会有细微的差别，可以被用来为不同的用户分配特定的编号. 使用 Canvas 技术在用户浏览器上进行隐秘的绘画，调用 *toDataURL*()方法将图片转化为 base64 编码的字符串，其中编码后字符串的最后一块是可作为用户唯一表示的 32b 的循环冗余校验码(cyclic redundancy check, CRC). 截至 2014 年，Acar 等人[7]调研发现在 Alexa 前 10 万的网站中已经有 5% 在使用 Canvas 指纹，且 Canvas 指纹追踪在网站的二级页面中比在网站的首页中可能更多[45]. Laperdrix 等人[46]也进行了大规模的 Canvas 指纹测试，论证了 Canvas 是最有区别性的指纹. 但 Canvas 指纹含有一定的碰撞率，Daud 等人[47]通过网站测试研究，论证了不同操作系统和相同机器规格下，Canvas 指纹识别值并不唯一. Canvas 指纹的生成不会引起用户的注意，因此其已较多应用于网络安全领域[48].

2)WebGL 指纹

WebGL 指纹和 Canvas 指纹原理基本相同. 通过 HTMLCanvasElement 元素可以获取到 WebGL 对象 *canvas.getContext*("*webgl*")，基于 WebGL 的 API 接口(例如：*WEBGL_debug_renderer_info*)获取用户的硬件信息，比如显卡名称、显卡型号、显卡制造商等. 只不过 3D 图像会和 WebGL 的某些属性值绑定在一起(比如显卡提供商、显卡型号、压缩等级等)拼接成很长的字符串，Hash 后获得一个缩减过的信息量，保留具有信息差的字符串作为 WebGL 指纹.

3)Audio 指纹

Englehardt 等人[49]发现了一种通过音频为设备生成指纹的方法，即音频(audio)指纹. 操作系统提供了专门处理音频的接口 AudioContext API，该 API 可以调用振荡器接口 OscillatorNode 来产生周期波形，然后调用信号处理模块将信号压缩，再通过实时频域和时域分析(time domain analysis)转换为频率域，最后生成指纹. 音频指纹与 Canvas 指纹类似，因浏览器和以及处理音频信号硬件的细微差别，产生不同的音频结果. 2016 年，Englehardt 等人[50]使用 OpenWPM 在 Alexa 前 100 万的网站中进行了测试，发现当前只有少数网站使用了音频指纹.

2.2 跨浏览器追踪

浏览器在接入网络时，都需要调用系统中的硬件、网络等信息获取网站内容资源并进行渲染. 获取系统和硬件层面特征的能力是所有浏览器的共性，可为攻击者提供更多获取设备信息的渠道，因此跨浏览器追踪也可称为设备追踪. 通过测量硬件特征、枚举缓存和网络信息，能够实现对用户的跨浏览器的标识和追踪[51]，也为指纹追踪提供了更多的稳定性.

2.2.1 基于硬件信息的设备追踪

设备硬件信息具有较强稳定性，尤其是硬件属性信息几乎不会发生改变. 不同设备的硬件在提供特定功能时，往往会输出特定的属性信息，以及不同的功性能表现. 通过获取或测量设备硬件的运行差异性，提取其较为稳定的特征项，作为设备唯一标识的特征信息.

1) GPU 特征信息

利用 WebGL 的 API 接口能够获取用户的硬件属性信息，比如显卡名称、显卡型号、显卡制造商等.

文献 [52] 基于设备的 GPU 时钟频率、GPU 时钟偏差来对设备进行 GPU 指纹识别实验, 论证了 GPU 的硬件特征不仅能够区分不同硬件配置的设备, 甚至能够区分具有完全相同硬件配置的设备.

2017 年, Cao 等人[53] 提出了一种依赖于设备操作系统和硬件功能的指纹识别技术, 使用 WebGL API 呈现 31 个不同的计算机图形任务(包括纹理、反锯齿、线条、光线、音频、视频、旋转、语言库等不同类型和角度), 然后从测试任务中提取信息 Hash 成指纹. 该指纹不仅支持单浏览器追踪, 还支持跨浏览器对用户进行追踪. 即使用户切换浏览器, 也能够唯一标识设备.

2) CPU 特征信息

利用对浏览器进行基准测试可以推断用户设备的 CPU 中是否存在 AES 指令集(aes-ni)和英特尔睿频加速技术(turbo boost)2 种扩展[54] 的理念, Saito 等人[55] 使用侧信道技术识别了 CPU 的内核数、超线程技术(hyper threading technology)可用性、SSE2 指令集(streaming SIMD extensions2)可用性、ASE-NI 可用性以及 CPU 系列类型等信息. Schwarz 等人[35] 利用由环境引起的浏览器引擎的细微差别, 自动化识别浏览器属性推断主机系统信息.

3)音频特征

频域在不同的浏览器中是不同的[53], 因此 Audio 指纹会受浏览器的影响, 不能完全反映出声卡的特征. 但是频率和峰值的比在浏览器之间是相对稳定的, 在频率和值的坐标系上创建一个间距很小的列表, 并将峰值频率和峰值映射到对应的表格项. 这种情形下的列表可用于跨浏览器的特征.

4)电池损耗度

Diaz 等人[56] 发现可利用火狐浏览器在 Linux 上提供的高精度读数发现用户的电池容量, 实现短时间内跟踪网络用户, 且旧电池容量或用过的电池的风险更高.

5)引擎性能监测

使用计时 API 函数获取指令序列的执行时间来计算硬件指纹, 能够在同类场景以及异构场景中区分完全相同硬件和软件的计算机集[57].

2.2.2 基于网络信息的设备追踪

IP 地址作为用户接入网络的“地址编号”, 是在对用户进行 Web 追踪时非常重要的信息. 设备接入 Web 所使用的公网出口 IP、内网 IP 和内网拓扑信息都有可能被用作用户追踪和分析的有效手段.

1) IP 地址及内网拓扑

Mishra 等人[58] 论证了公网 IP 地址仍然是在线跟踪的主要载体, 87% 的参与者至少保留 1 个 IP 地址 1 个月以上. 文献 [59−60] 基于网页即时通信(web real-time communication, WebRTC)可获取内网 IP 地址的漏洞[61], 提出了基于浏览器网络扫描器 BrowserScaner, 依据网络响应信息判断内网拓扑和内网主机存活情况, 甚至能够探测活动节点的开放端口.

2)TLS 会话

Sy 等人[62] 提出了一种可以将跟踪周期延长到会话恢复机制的生命周期之外的“延长攻击”, 证实了通过 TLS 会话恢复能够跟踪用户平均长达 8 天的时间. 而 TLS1.3 草案使得至少 65% 的用户可由至少一个网站永久跟踪.

2.2.3 基于缓存信息的设备追踪

对缓存中的敏感地理位置实行推断攻击[63], 可跟踪用户所处的国家和城市. Klein 等人[64] 利用可控的域名解析服务器(domain name system, DNS), 为每个用户分配唯一的 DNS 解析记录, 基于用户主机的 DNS 缓存机制, 实现对用户的追踪, 克服浏览器的“隐私模式”. 作为 Internet 的重要组成部分, 许多网络应用提供商使用内容分发网络(content delivery network, CDN)部署的大量缓存代理, 因此能够使攻击者获得缓存数据的未授权访问的 Web 缓存欺骗[65], 使大量网络用户处于危险之中.

2.3 跨设备追踪

跨设备追踪(cross device tracking, CDT)[66] 分为确定性追踪和概率性追踪, 确定性跨设备追踪主要通过邮箱、手机号等登录账户实现跨设备追踪; 概率性跨设备追踪则借鉴和分析了用户历史记录、行为等信息, 实现统计学上的关联. 本文只讨论概率性的跨设备追踪.

2.3.1 跨设备特征分析

当前的追踪手段往往使用多种技术相结合进行综合性的用户追踪, 且移动端对 HTML5 的支持相较于桌面客户端更为迅速且全面, 因此用于指纹追踪的特征绝大部分可同样在移动端获取.

文献 [46,67] 对桌面设备和移动端设备进行了指纹研究, 虽然移动端缺失了字体和插件 2 个较为有区别性的特征, 但依然具备 81% 的指纹独特性, 并且 HTTP 标头和 HTML5 的 Canvas 指纹[59,67] 在识别这些设备的浏览器方面起着至关重要的作用. 文献 [68] 证实了大部分网站与第三方网站定期收集的 Cookie、IP、位置、行为数据等信息相同, 而这些信息可允许第三方将多个设备与持续性的标识符关联起来.

Kane 等人[69]研究了用户在不同设备之间的信息共享方式和访问网页活动的重叠度，论证了信息共享的便捷性可提高移动网络的可用性.

Karakaya 等人[70]总结了用于概率跨设备定位的常见物理和行为特征，分析了 16 个跨设备追踪的学术研究，其认为对行为模式的探索是实现概率跨设备跟踪的关键. 例如常用的属性是 URL 地址，以及通过该 URL 访问的内容或访问的网站和应用程序的上下文信息. 从站点内容或上下文分类中提取的自然语言处理（natural language processing, NLP）特征对跨设备匹配非常有用.

2.3.2 跨设备追踪方案

我们对常见标识符进行了大规模探索，并量化了它们揭示的主机识别信息的数量.

使用来自 Hotmail 和 Bing 的长达 1 个月的数据集，Yen 等人[71]验证了通用标识符（包括用户代理、IP、Cookie、登录 ID 等）可以帮助以高精度和高召回率跟踪主机. 通过跨主机聚合信息，发现了超过 75 000 个将 Cookie 转发到分布式位置的机器人账户相关的隐蔽恶意攻击. Zimmeck 等人[72]展示了一种检测跨设备跟踪发生的方法，基于从 126 位互联网用户收集的跨设备跟踪数据集，探索并论证了用户设备的 IP 地址和互联网历史记录的相似性能够将移动设备连接到桌面设备的数据集中，其匹配率为 0.91. Díaz-Morales[73]提出了一种基于半监督机器学习方法来处理用户跨设备识别的解决方案，以识别哪些 Cookie 属于使用设备的个人.

2.4 指纹关联研究

指纹动态变化原因可分为 3 大类：浏览器或操作系统更新、用户操作更新和环境更新[74]. 指纹的不稳定性能够通过选取较为稳定的特征来进行缓解和弥补，但依然不能实现对用户长期追踪. 因此研究学者设计了指纹关联算法作为另一个使追踪更加持久化的手段.

当前的指纹关联研究均使用相似度分析对指纹进行关联度判定，也用以证明持续追踪的可能性[75]. 其中文献 [76] 引入了指纹的辅助属性判定，使用专家打分法和编辑距离计算相似度，文献 [77] 在此基础上提出了一种高性能的指纹识别模型，将存储型追踪标识符作为指纹关联的重要依据. 文献 [78] 通过实验证明了计算机在运行特定程序时产生的不同的 CPU 图形，可在指纹关联时辅助解决其他属性易被伪装的问题. 文献 [79] 提出了使用规则和算法进行关联的 FP-STALKER，其可追踪浏览器 54.48 天，并且 26% 的浏览器能被追踪超过 100 天. 文献 [80] 针对安卓（Android）设备中的指纹进行了研究，提出了基于双向循环神经网络（recurrent neural network, RNN）的指纹识别模型.

但是关于指纹中特征组合和识别精度之间的相关性研究较少，文献 [81] 使用了在网站上收集的 9 457 个指纹样例，来计算不同特征组合的识别精度. 与电子前沿基金会的结果相比，在 Windows, Mac, iOS, Android 系统上的测试，分别实现了 1.006%, 3.894%, 7.456%, 3.840% 的识别精度提升.

2.5 追踪分析讨论

存储型追踪技术可以服务于所有范围的追踪，关联算法需要定向依据追踪技术的不同进行算法的改进和升级. 按照追踪的范围，发现跨设备追踪和跨浏览器追踪所使用的特征均可应用于单浏览器追踪. 跨浏览器追踪层面较多使用设备硬件等稳定且不易变化的特征，跨设备追踪本身的特殊性无法使用稳定的设备特征进行关联，但具有较多用户个人特色的历史记录、交互性特征可作为对同一用户不同设备进行分析和关联的重要特征.

一些研究学者对现有的指纹追踪进行检测[82-84]，进行了大规模分析[85]，评估其在全球范围内的使用情况，当前某些指纹追踪技术应用较为广泛，但是在实际生活中，较多的网站厂商使用一种或多种指纹追踪技术，隐秘地进行大数据杀熟、价格歧视[86]等操作. 且指纹可能被攻击者恶意利用，基于指纹的某些特征获取技术实施攻击和漏洞利用，窃取用户敏感信息. 这在一定程度上为用户造成了极大的隐私安全风险.

3 Web 追踪防御技术

Web 追踪防御技术是用户隐私保护大趋势下的必然发展方向. 传统的用户自我保护措施主要是禁用 Cookie，设置 DNT 选项，使用虚拟专用网络（virtual private network, VPN）或 HTTP 代理、隐身浏览模式、隐私保护类扩展等方式来阻止广告追踪. 但这些方式依然无法阻挡指纹中多种多样的特征获取和追踪. 近些年的 Web 追踪防御技术大多集中在与指纹型追踪的对抗方面，本文按照防御技术的形态对当前研究进行了分类，并对使用的抵抗手段进行对比.

3.1 扩展防御

基于浏览器的扩展中，广告拦截器和内容拦截器比跟踪器拦截器的使用更广泛[87]. 且使用安装扩展

形式的防御技术,能够在不影响用户整体使用习惯和环境的前提下,增加对 Web 追踪的防御能力.

早期的追踪防御技术大多着眼于隐藏和修改敏感信息,即通过干扰真实信息的方式来避免追踪. 例如 TrackMeOrNot[88] 扩展提供了与供应商共享足迹的选择权,依据用户指定的隐私要求有选择地向供应商披露自身活动. Torres 等人[89] 开发了一个原型工具 FP-Block,在允许常规追踪的前提下,对抗基于跨域指纹的跟踪,确保任何追踪者在每个站点上都会看到用户不同且不相关的指纹,使指纹追踪无法再在网络上对用户进行持续性跟踪. Al-Fannah 等人[90] 发现大约有 69% 的网站可能涉及第一方或第三方浏览器指纹识别,且第三方浏览器指纹识别占据了很大比例,因此设计了 FingerprintAlert 来检测(可选)并阻止所访问网站的指纹尝试.

针对扩展信息泄露问题上,Latex Gloves[26] 基于 Web 代理在网页和扩展之间进行控制,使用白名单的方式精细化地设置允许将哪些扩展程序加载到页面中,以应对使用 WAR 探测和 ID 枚举进行扩展识别. 但白名单的获取目前没有官方渠道,需通过用户自行判断设置. Starov 等人[91] 设计并开发了浏览器扩展程序 BrowsingFog,通过不断评估用户的浏览习惯并模拟对随机类别网站网页的访问,从而将用户的真正兴趣隐藏在不相关信息的“雾化”中. 这是一种假设用户存在 1 个或多个浏览数据泄露扩展的情况下保护用户隐私的方法.

Canvas Blocker[92] 是防止 Canvas 指纹的 Firefox 扩展,通过更改使用 Canvas 2d, WebGL, Audio, Screen 等 API 获取信息的值,欺骗追踪者对真实信息的获取. 除此之外,还使用了黑白名单机制对网站追踪进行允许和拦截. Canvas Deceiver[93] 也是抵抗 Canvas 指纹的浏览器扩展,在网站发出请求调用画布指纹识别 JavaScript 文件时,拦截请求并提供脚本的修改版本,修改版本的脚本行为方式与原始版本相同,但使用了一个持久的、预先确定的字符串欺骗使用 Canvas 指纹的追踪者.

Leiva 等人[94] 提出了一种对抗性噪声方法,减轻利用鼠标光标大规模捕获用户行为数据,来跟踪预测人口统计变量的用户分析技术,并发布了数据和实现对抗性方法的 Web 浏览器扩展.

3.2 浏览器内嵌防御

现有防御技术存在修改现有浏览器,通过在浏览器中内嵌防御机制提升浏览器本身的防护能力的方式,提供更深层且更加全面的防护效果,我们称之为“浏览器内嵌防御”. 内嵌防御着眼于缓解或拦截特征信息的获取渠道,或防止指纹被关联,破坏持续追踪.

TrackingFree[95] 是第 1 个反跟踪缓解唯一标识符的浏览器. 其没有禁用唯一标识符,而是将它们隔离到不同的浏览器主体中,使得标识符在不同网站之间不是唯一的,从根本上切断第三方 Web 跟踪的跟踪链. Laperdrix 等人[96] 开发了一个名为 Blink 的应用程序原型,探索了软件组件本身的多样性,利用虚拟化和模块化体系结构在多个级别自动组装和重新配置软件组件,修改平台指纹中展示的指纹,实现针对浏览器的移动目标防御指纹追踪. 经过在数千个平台中的测试和验证,Blink 可使用户展示不同的指纹,甚至欺骗商业性的指纹脚本. Baumann 等人[97] 演示的防指纹策略可在不停用 Flash 指纹的情况下对其进行保护,并提供稳健且不易检测的反画布指纹(anti-canvas fingerprinting),在不使浏览器失去可用性的前提下,使用大量的真实信息来隐藏系统和浏览器属性. Yu 等人[98] 提出了一种基于 k 匿名(k-anonymity)方法的新技术,识别有可能唯一地标识单个用户的数据元素,并将其从请求中删除. 通过使用 Cliqz 浏览器和 Cliqz 扩展程序进行大规模在线跟踪研究,发现其中所访问页面的 95% 包含对潜在跟踪者的第三方请求,而 78% 试图传输不安全的数据,单个追踪最多可以达到德国所有页面访问的 42%. 随后研究团队发布了对跟踪请求中用户标识符的算法检测[99],使用了 2 周内来自 20 万德国用户,涵盖了超过 350 000 个不同站点的 2 100 万次页面访问,结果彰显出了不同地区之间的追踪差异性,以及追踪厂商在不同地区市场上的差异. 同时也证明了谷歌和 Facebook 无处不在.

Starov 等人[100] 将完全不需要扩展功能的页面修改称为扩展膨胀. 经过分析来自 Google Chrome 商店的 58 034 个扩展,发现有 5.7% 的扩展膨胀是不必要的,且这些扩展中的 61% 是唯一的,可以被滥用进行指纹识别. 为了保护用户免受由于膨胀引起的不必要的扩展指纹侵害,Starov 直接修改 Chromium,设计和实现了一个浏览器内的机制,为所有网站中的扩展提供粗粒度的访问控制. 该机制使用 Chrome 内置的企业策略特性,不仅可以保护用户不被指纹识别,还可以防止恶意扩展程序从敏感网站窃取用户数据.

FPGuard[101] 是一种运行时指纹检测和预防方法,包含检测和预防 2 个阶段. 首先通过 9 种度量标准来检测网站中与指纹相关的活动,计算可疑分数,如果

该分数超过特定阈值，将网页标记为指纹识别器，并将网站 URL 以及对应的度量标准存储到黑名单数据库中. 然后使用包含 4 个核心引擎（objectRand, pluginRand, CanvasRand, fontRand）的随机化技术，对属性值进行随机化修改. 例如 objectRand 引擎在运行时生成随机对象实现对象属性随机化，pluginRand 引擎将不存在的虚拟 Plugin 和 MimeType 对象添加到浏览器的当前插件和 mimeTypes 列表中，以更改列表的顺序. FPGuard 结合了随机化和过滤技术，在用户每次访问网页时更改指纹，这种方法可以有效地识别和缓解与指纹相关的活动，并将正常网页与异常的含有指纹追踪的网页区分开.

破坏指纹的可链接性可以防止指纹的关联，即阻止多次访问的指纹链接同一指纹的能力，破坏指纹的稳定性后，追踪器也难以跨浏览会话关联指纹. 基于此 Nikiforakis 等人[102]提出了利用随机化破坏可链接性来消除无状态跟踪的 PriVaricator. 通过评估几个系列的随机函数来寻找可以在指纹预防和破坏现有站点之间达到最佳平衡的函数，进而集中对字体和插件相关的属性进行随机化. 在没有明显开销的情况下，可有效防止指纹的追踪，且对 Alexa 前 1 000 个站点的内容造成的损害也最小.

Laperdrix 等人[103]提出了基于 Firefox 的修改版 FPRandom，通过“破坏”随时间推移的指纹稳定性减轻指纹追踪的风险. FPRandom 在 Firefox 浏览器功能的计算中增加了随机性，为每个浏览会话提供略有不同的结果，其主要针对 Canvas 指纹、Audio 指纹、通过 JavaScript 属性顺序对浏览器进行追踪的指纹. 主要缓解方法有：1）在 CanvasRendering Context2D 类的 ParseColor 函数内部引入随机性，在 SetFont 函数中将脚本设置的字体更改为操作系统中的字体；2）在 AudioContext 模块的 AudioNodeEngine 的 AudioBuffers 的核心中引入微小的噪声，使指纹脚本产生不同的 Hash 值；3）修改 Firefox 的 jsiter 类，激活浏览器中的特殊标志“JS MORE DETERMINISTIC”，通过在每次启动浏览器时生成唯一的枚举顺序来缓解指纹追踪.

3.3 防御框架、工具或机制

立足于用户的设备和浏览器的差异性，除重写指纹值外[104]，一些研究学者设计了一些更高层次、更重型的防御技术，例如构建防御框架、进行属性重组和选择的工具或者一些深层的防御机制，来有效地防御指纹追踪.

基于 WebGL 指纹识别中的渲染差异是由浮点操作引起的，因此 Wu 等人[105]提出了一种称为 UNIGL 的新型系统. UNIGL 重新定义了所有浮点操作，显式地用 OpenGL 着色语言（OpenGL shading language, GLSL）程序编写或由 WebGL 隐式调用这些浮点操作，实现了重写 GLSL 程序并在现有 WebGL 功能的支持下制作统一的 WebGL 渲染过程，来减轻指纹识别因素的影响. 经过众包测试评估表明，UNIGL 可以通过合理的帧率（frames per second, FPS）防止基于 WebGL 的最新指纹.

Jueckstock 等人[106]编写 1 个少于 600 行的补丁，完成了一个托管在 Chrome 浏览器的 JavaScript 引擎（V8）内部的动态分析框架 VisibleV8. VisibleV8 保存了网站使用 JavaScript 的所有数据，并为网站创建“行为配置文件”，研究人员可使用该配置文件以及所有支持数据来识别网站中使用 JavaScript 破坏 Web 浏览器和用户信息的行为. VisibleV8 隐藏在 JavaScript 本身的引擎中，其性能优于同等内嵌式的安装模式，并且拦截了内嵌安装无法阻止的访问. 这种全面的覆盖模式可以隔离和识别使用 JavaScript 代码检测 46 个 JavaScript 命名空间和浏览器平台. 在补丁中只有 67 行修改了 V8 的现有行为，因此 VisibleV8 是非常轻量且易于维护的.

FPSelect[107]是一个属性选择框架，允许验证者调整指纹探针以进行 Web 身份验证，这种方法使用指纹的成本较低，并且达到了针对字典攻击的最低安全级别. 其根本是将问题转化为搜索满足安全要求并最小化可用性成本的属性集. 通过对其安全性和可用性进行比较，FPSelect 生成的指纹属性集大小减小了 97 倍，收集指纹的速度提高了将近 3 361 倍，而 2 次观测之间的属性变化平均减少达 7.2 倍.

随后 Andriamilanto 等人[108]开发了一种属性选择工具 BrFAST. 它搭载了 FPSelect 算法，从数百个可用属性中选择要实施的属性，根据安全性和可用性之间的权衡来严格选择浏览器指纹属性. BrFAST 还可以扩展为使用其他参数、其他属性选择方法，其他安全性和可用性度量或其他指纹数据集，有助于在搜索要使用的最佳属性时可视化对可能性的探索，比较属性集的属性以及比较几种属性选择方法.

当前研究针对跨设备的防御较少，2018 年，Solomos 等人[109]设计了一种通过模拟最终用户的真实浏览活动来触发 CDT 的方法，然后利用先进的机器学习工具对其进行检测和测量. 随后 Solomos 等人[110]设计构建了一个 CDT 测量框架——TALON，基于模拟用户在不同设备上的真实浏览活动来触发和检测跨设备定向广告，检测 CDT 并测量影响其性能

的因素. 多项关键实验证明, 该框架能够检测和测量平均 ROC(receiver operating characteristic)曲线下的面积(area under curve, AUC) 得分为 0.78~0.96 的 CDT, 还能够提供有关 CDT 实体行为和对用户隐私影响的重要见解. TALON 能够成为隐私研究人员、政策制定者、用户的重要工具, 来提高广告生态系统使用的跟踪实践的意识和透明度.

3.4 防御对策或环境

指纹型追踪在用户正常网站会话中以静默方式获取属性的组合来识别浏览器, 从而确定该浏览器的用户. 与之前的在指纹信息获取后进行阻止或欺骗的防御方式不同, 构建一种防御机制或者用户使用环境, 能够在浏览器特征被获取的层面进行信息的统一化或者随机化, 即从信息产生的源头进行防御.

CloakX[111] 是一个基于客户端的防指纹对策. 通过在客户端重写扩展名且以与扩展名无关的方式运行, 使客户端信息多样化. 可以防止使用锚定标记(由任何网页均可直接访问的工件组成的指纹)进行扩展检测, 并降低使用结构指纹(根据扩展程序行为构建的指纹)进行扩展检测的准确性. 经过大规模自动化分析和深入的手动测试, 评估 CloakX 在 18 937 个扩展名上的功效以及启用 CloakX 的扩展的性能, 证明了 CloakX 原型可以成功处理大多数浏览器扩展.

Gómez-Boix 等人[112] 提出一种缓解指纹追踪对浏览器配置及其运行环境进行信息收集的方法. 先使用算法识别易于共享相同或相似指纹的设备, 并为其提供新的非唯一指纹. 然后在虚拟化的 docker 容器中, 使用该指纹组装和运行 Web 浏览器, 最终使得同一个集群的所有设备均运行使用同一个指纹的 Web 浏览器. 通过为用户提供一个通用的共享指纹的配置, 最大程度地减少设备的可识别性和 Web 浏览器配置中的更改次数.

Web 追踪防御技术的飞速发展是在信息爆炸式发展、用户隐私意识增强的社会条件下的必然发展趋势. 如表 5 所示, 不同类型的防御方式也各自存在其自身的局限性. 例如用户对不同防御形态的接受度、防御实现技术对 Web 生态造成的影响、可能存在的对用户体验度的影响等. Web 追踪和防御是相辅相成的技术, 如何实现一个平衡, 也是需要面对和解决的问题.

4 面临的挑战与展望

Web 追踪中指纹追踪技术的多样化在一定程度上满足了 Web 应用厂商的需求, 但用户对于安全隐私的诉求也极大推动着浏览器指纹防御技术的快速发展. 本文从 Web 追踪和防御技术 2 方面分别进行剖析, 分析当前 Web 追踪的现状, 以及面临的挑战与未来可能的发展方向.

4.1 追踪技术分析

针对使用浏览器特征的指纹追踪研究(见第 2 节)中, 包含具有较强用户特点的历史记录、扩展等类型的特征, 以及依赖硬件属性和硬件功性能的较为稳定的设备特征. 但多数的指纹追踪结合使用多种不同维度的特征来提高指纹的唯一性[115].

1) 追踪唯一性讨论

Eckersley 等人[1] 在 2 周内收集了含有 8 个特征的 470 161 个指纹, 其中 83.6% 的指纹是唯一的, 且在开启了 Flash 和 Java 的情况下, 唯一性能够达到 94.2%. Laperdrix 等人[46] 使用 AmIUnique 网站在一年中收集了包含 17 个特征的 118 934 条指纹, 得出 89.4% 的指纹是唯一的, 并论证了 Canvas 指纹的稳定性. Gómez-Boix 等人[67] 在法国排名前 15 的一个商业网站中收集了 1 816 776 个普通用户的电脑端的浏览器指纹, 经过评估得出, 数据集中只有 35.7% 的唯一指纹, 远低于过去报告的百分比.

Eckersley[1] 和 AmIUnique 这 2 项大规模的数据统计, 均是在通知用户了解使用指纹追踪的前提下进行收集的, 其指纹的用户来源对在线隐私了解较多, 相较于普通用户的操作更为谨慎, 而文献 [67] 的指纹来源为网站中的普通用户. 通过数据对比分析, 用于追踪的网站、使用的特征、地区等差异可能对数据的影响较小, 但指纹收集的目标人群的差异可能会导致指纹唯一识别的有效性发生较大变化. 虽然指纹特征的信息熵能够识别足够多的用户, 但在实际应用场景中, 普通用户群体中指纹的唯一性远比对隐私敏感的用户低.

2) 追踪持久性讨论

虽然指纹经常变化, 但依然可以在一段时间内跟踪相当一部分浏览器[116], 指纹自身的稳定性是能够实现持久追踪的基本要素, 而关联算法是能够实现持久追踪的关键因素. 不易变化的硬件特征信息能够有效地为指纹添加稳定性, 并且可以在关联算法中作为相似性评估的重要依据. 当前的指纹追踪技术中, 仅有少量的研究对追踪持久性[117] 进行讨论, 在指纹关联分析方面的研究较为欠缺.

部分研究学者着重于从跨浏览器和跨设备的角度实现指纹追踪, 增强用户追踪的持久性. 确定性的

Table 5 The Forms and Used Resistance Means of Defense Technology

表 5 防御技术的形态与其使用的抵抗手段

防御技术	形态				抵抗手段								
	扩展	内嵌防御机制	框架/工具/机制	对策或环境	随机化策略	引入噪音	属性重组	重写/欺骗	访问控制	同质环境	算法	信息隔离	用户参与
FPGuard[101]		√			√			√					
PriVaricator[102]		√			√			√					
Blink[96]		√					√	√					
FP-Block[89]	√				√							√	
TrackingFree[95]		√										√	
Cliqz[98]		√						√			√		√
TrackMeOrNot[88]	√								√				
隐藏特征属性[97]		√			√								
Tor[113]		√		√	√			√	√	√	√		
BrowsingFog[91]	√					√							
FPRandom[103]		√			√	√			√				
Latex Gloves[27]	√												√
PETInspector[114]				√				√					
FingerprintAlert[90]	√								√				
抑制扩展膨胀[100]		√							√				
CloakX[111]				√				√					
UNIGL[105]			√					√					
docker 集群组装[112]				√			√			√	√		
VisibleV8[106]			√						√			√	
Canvas Blocker[92]	√							√	√				
Canvas Deceiver[93]	√							√					
FPSelect[106]			√				√				√		
BrFAST[108]			√				√				√		
My Rules[94]	√					√							

注：“√”表示防御技术具备某种形态或包含某种抵抗手段.

跨设备追踪，较多依赖于用户的登录信息，因此能够具备较强的稳定性和关联性；而现有的概率性跨设备追踪的分析较多依赖于用户历史记录等不稳定的特征，用于关联分析的普适性较低，因此概率性的跨设备追踪在结合 Cookie 类存储标识的前提下会更加有效和稳定.

Web 追踪防御技术可能会使浏览器更容易被识别[118]. 发现指纹中可能被修改的特征，将其还原成真实的特征值，加强有效的定位和追踪也是未来可能的研究方向. 例如测试套件 FP-Scanner[119]，能够探索浏览器中指纹的不一致性，检测潜在的更改，揭示改变的指纹属性的原始值. 如果这一结果可以被指纹识别技术利用，就能够更准确地定位使用防御技术的浏览器.

3）挑战与展望

通过大规模的数据研究分析，可以得出指纹追踪技术的唯一性与其数据源有极强的关联关系，Web 追踪领域需要一个或多个较大的指纹数据集，用于分析指纹追踪的有效性，以及关联分析研究. 在指纹追踪过程中，需要在保证指纹唯一性的同时对特征进行评估和择优选取，避免收集过多特征对数据传输和存储造成的压力.

针对指纹追踪的特征进行研究，尤其是含有较多用户习惯和交互式的特征，可为不同的用户构建稳定且全面的用户画像，或辅助 Web 安全领域针对恶意数据进行分析，以及对恶意行为的检测和追踪溯源等. 比如双重身份认证[120-121] 和账户安全检测[122-123]、检测爬虫或者 Web bot[124-125]、党派分析[126] 等. 或者辅助安全研究，从访问记录中提取危险路径分析恶意 URL 的 Web 访问路径[127]，将指纹应用到对攻击者的诱捕和欺骗中，实现主动的追踪溯源[128]，协助网络攻击取证[129-130].

4.2 防御技术分析

尽管当前已经出现各种安装形态的、各种不同功能的指纹防御技术,但依然存在各自的劣势.洋葱浏览器(Tor)[131]当前在匿名网络中被广泛使用,用以从多个方面防御各种不同类型的追踪技术,保护用户的隐私.如表6所示,本文针对不同指纹防御所适用的技术进行了归类分析和概括,并结合Tor浏览器进行对比分析,指出了当前防御技术的挑战和方向.

Table 6 The Defensive Coverage that Defensive Technology Possessed to Tracking Technology

表6 防御技术对追踪技术的防御覆盖度

防御技术	基本特征	IP	Cookie	Canvas指纹	WebGL指纹	Audio指纹	历史记录	字体	扩展	缓存	硬件信息	JS属性枚举检测	TLS会话	交互式追踪
FPGuard[101]				√				√				√		
PriVaricator[102]								√						
Blink[96]	√			√		√					√			
FP-Block[89]														
TrackingFree[95]														
Cliqz[98]	√	√	√											
TrackMeOrNot[88]							√							
隐藏特征属性[97]				√										
Tor[113]	√	√	√	√	√	√								
BrowsingFog[91]									√					
FPRandom[103]				√		√						√		
Latex Gloves[27]									√					
PETInspector[114]	√													
FingerprintAlert[90]	√		√											
抑制扩展膨胀[100]									√					
CloakX[111]									√					
UNIGL[105]					√									
docker集群组装[112]		√		√	√	√					√			
VisibleV8[106]												√		
Canvas Blocker[92]				√	√	√								
Canvas Deceiver[93]				√										
FPSelect[106]	√													
BrFAST[108]	√													
My Rules[94]														√

注:“√”表示防御技术覆盖了该项追踪技术.

1)防御形态分析

扩展形态的防御较为轻量级,安装和使用较为便捷,多数扩展的功能集中于抵御某项单一的指纹追踪方式,例如防御当前应用较为广泛的Canvas指纹、历史记录等.但这种类型的防御机制无法抵御使用多种特征追踪的指纹模式,即便某个特征值不一致或者缺失,网站追踪者仍可通过多重追踪技术,并使用相关算法将属于同一用户的指纹关联起来.

浏览器内嵌防御利用随机化、添加噪音、函数重写等方式修改内嵌的函数实现机制,防止追踪脚本获取到真正的属性特征值.尽管为特定的单个属性或属性集提供人为或随机值可以至少部分成功地干扰指纹识别的操作,但这种方法也可能产生不利影响[132].这种形式需要用户以更换浏览器为代价,达到对单个或多个特征的追踪技术的防御,虽然在防御性能和效果上优于外在的扩展和工具框架等策略,但防御型浏览器如未经商业化加持的情况下,在推广和应用上面临着巨大的困难:一方面是Tor浏览器防御能力较强,并占据着绝大部分的市场;另一方面,用户对“被改装”过的浏览器接受度可能不够高.

工具和框架形态的防御,从独立于浏览器之外的角度出发,搜索配置选择进行展示的最佳属性,混

淆追踪者获取的信息. 利用属性重组选择特征值的防御技术, 一定会以消耗时间和减弱性能为代价, 且随着指纹特征的增多, 会导致浏览器中出现较多的异常组合[133].

构建防御对策或环境是从技术源头缓解指纹追踪的方式. 例如通过使用同质化环境可以从指纹产生的源头消除其唯一性, 重写扩展名和更改扩展运行方式可以有效抵抗使用扩展 ID 和 WAR 对扩展的枚举. 但构建同质化环境和对扩展进行处理和运行的方式偏向于理想的防御方式, 无法广泛地应用于实践.

2) 防御技术的对比分析

随机化和引入噪音是最常见且易于实现的防御技术, 能够通过欺骗或者扰乱信息的方式, 削弱实际设备和浏览器的独特性和唯一性, 降低被追踪的可能. 属性重组、重写指纹值等技术的本质也是通过虚假或冗余的信息降低用户的可识别性. 这些技术形式都在追踪系统中生产了大量的虚假无效信息, 消耗和占用服务资源. 这类数据可能会影响未来的 Web 追踪生态, 对未来的技术发展造成不利的影响.

通过监控指纹追踪的属性获取情况, 了解和控制可能被泄露或用来追踪的信息的输出和运行方式具有较好的实践性. 例如控制既定对象的枚举顺序、浏览器扩展在不同网站中的运行方式、可能被泄露的指纹信息等. 但无法做到全面和综合性的防御, 对新追踪方式的自适应性不强.

通过一定的手段, 对不同的访问网站实现信息隔离, 使多个不同的追踪网站获取到的指纹信息各不相同. 这一方案能够较好地组织不同网站之间的追踪信息共享, 但对同一网站的持续性追踪具有较低的防御力.

用户介入和参与的防御形式, 虽然将选择权交到了用户手中, 但绝大部分的网络用户没有较多的技术理解和专业知识, 无法做出较为有利的判断. 且从长期来看, 用户参与过多会降低良好的体验度.

3) 挑战与展望

当前有诸如欧盟相关法律为加强隐私做出了努力, 例如第三方跨设备跟踪的公司将被要求在其隐私政策中明确数据会被跨设备使用, 但违反行为还是很普遍且公然的[134], 因此依然需要更有效的工具和框架来进行合规性监视和执行. Luangmaneerote 等人[133] 通过将所有的指纹防御技术应用到浏览器中, 访问开发的混合指纹网站, 了解指纹抵抗效率以及指纹防御对用户或网站造成的影响, 结果表明, 如今所有的防御对策, 都无法阻止各种指纹追踪[135], Tor 依然是当下最有效的指纹防御技术[107].

本文依据当前防御技术的能力, 对能够抵御的追踪技术进行了整理和总结, 如表 6 所示. 现有的防御技术大多能够实现对浏览器基本特征信息获取和主流指纹追踪技术的防御. 但是在缓解系统层面的缓存追踪、TSL 会话追踪, 以及硬件追踪方面表现较差. 最后我们从表 6 中可以得出与文献 [135] 同样的结果: 当前的追踪策略均无法完整且有效地对指纹信息进行防御.

基于以上分析可以得出, 对追踪防御的构建和实施, 必须了解指纹追踪的技术原理, 通过阻断其获取信息的渠道和可能存在的操作, 或者使用信息干扰和隔离的方式扰乱攻击者对信息源的唯一确认. 防御技术的形态和最终形式也可能为指纹追踪提供新的特征识别线索. 当前而言, 实现指纹防御的最佳状态, 就像在“严格”的 Tor 浏览器模式和正常的浏览器功能之间取一个中和的方法, 在不影响用户浏览体验的前提下, 尽可能地将指纹信息一致化, 且在每次启动浏览器或会话的时候, 将 IP 等特殊的特征随机化.

5 总　　结

在 Web 应用厂商对大规模用户群体的追踪定位需求, 以及用户对隐私敏感度提升的双向作用下, Web 的追踪防御技术也得到了迅猛发展. 本文首先对 Web 追踪中不同的追踪技术和发展形势进行了概述, 其次着重针对 2010—2021 年间 Web 追踪技术和防御技术进行了研究和概述. 以追踪的范围划分追踪层次, 介绍了特征的特点和获取方式, 与此同时描述了指纹关联技术, 并论述了特征与追踪范围的关系. 随后我们列出了指纹防御技术的研究热点, 指出了当前防御技术的 4 种不同形态. 针对 Web 追踪技术, 本文从唯一性和持续性 2 方面对追踪技术进行讨论, 从实现形态和技术特点对追踪防御技术进行分析, 基于对浏览器指纹追踪对抗技术的了解, 我们针对指纹追踪和防御 2 个方面提出了面临的挑战与未来的展望.

作者贡献声明: 王晓茜负责论文的撰写; 刘奇旭梳理论文框架和指导论文的撰写; 刘潮歌、张方娇和刘心宇收集和整理论文文献; 崔翔负责论文的整体规划。

参 考 文 献

[1] Eckersley P. How unique is your web browser? [C/OL] //Proc of the 10th Int Symp on Privacy Enhancing Technologies. Berlin: Springer, 2010[2021-03-21].https://link.springer.com/content/pdf/10.1007/978-3-642-14527-8.pdf

[2] Zhang Yuqing, Wu Qianru, Liu Qixu, et al. Research on security of third-party tracking[J]. Journal on Communications, 2014, 35(9): 1−11 (in Chinese)
(张玉清, 武倩如, 刘奇旭, 等. 第三方追踪的安全研究[J]. 通信学报, 2014, 35(9): 1−11)

[3] Bujlow T, Carela-Español V, Sole-Pareta J, et al. A survey on web tracking: Mechanisms, implications, and defenses[J]. Proceedings of the IEEE, 2017, 105(8): 1476−510

[4] Takasu K, Saito T, Yamada T, et al. A survey of hardware features in modern browsers[C] //Proc of the 9th Int Conf on Innovative Mobile and Internet Services in Ubiquitous Computing. Piscataway, NJ: IEEE, 2015: 520−524

[5] Soltani A, Canty S, Mayo Q, et al. Flash cookies and privacy[C] // Proc of the 2010 AAAI Spring Symp Series. Palo Alto: AAAI, CA: 2010: 22−24

[6] Samyk. EverCookie[CP/OL]. (2017-11-13) [2021-04-20]. https://samy.pl/evercookie/

[7] Acar G, Eubank C, Englehardt S, et al. The web never forgets: Persistent tracking mechanisms in the wild[C] //Proc of the 14th ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2014: 674−689

[8] West W, Pulimood S M. Analysis of privacy and security in HTML5 Web Storage[J]. Journal of Computing Sciences in Colleges, 2012, 27(3): 80−87

[9] Kimak S, Ellman J. The role of HTML5 IndexedDB, the past, present and future[C] //Proc of the 10th Int Conf for Internet Technology and Secured Transactions (ICITST). Piscataway, NJ: IEEE, 2015: 379−383

[10] Zhang Yuqing, Jia Yan, Lei Kenan, et al. Survey of HTML5 new features security[J]. Journal of Computer Research and Development, 2016, 53(10): 2163−2172 (in Chinese)
(张玉清, 贾岩, 雷柯楠, 等. HTML5新特性安全研究综述[J]. 计算机研究与发展, 2016, 53(10): 2163−2172)

[11] Nair K V, RoseLalson E. The unique ID's you can't delete: Browser fingerprints[C/OL] //Proc of the Int Conf on Emerging Trends and Innovations in Engineering and Technological Research (ICETIETR). Piscataway, NJ: IEEE, 2018 [2021-05-04]. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8529040

[12] Laperdrix P, Bielova N, Baudry B, et al. Browser fingerprinting: A survey[J]. ACM Transactions on the Web, 2020, 14(2): 1−33

[13] Felten E W, Schneider M A. Timing attacks on web privacy[C] //Proc of the 7th ACM Conf on Computer and Communications Security. New York: ACM, 2000: 25−32

[14] Weinberg Z, Chen E Y, Jayaraman P R, et al. I still know what you visited last summer: Leaking browsing history via user interaction and side channel attacks[C]//Proc of the 32nd IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2011: 147−161

[15] Wondracek G, Holz T, Kirda E, et al. A practical attack to de-anonymize social network users[C] //Proc of the 31st IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2010: 223−238

[16] Baron L D. Preventing attacks on a user's history through CSS: Visited selectors[EB/OL]. 2010[2020-05-11]. https://dbaron.org/mozilla/visited-privacy

[17] Janc A, Olejnik L. Web browser history detection as a real-world privacy threat[C] //Proc of the 15th European Symp on Research in Computer Security. Berlin: Springer, 2010: 215−231

[18] Olejnik L, Castelluccia C, Janc A. Why johnny can't browse in peace: On the uniqueness of web browsing history patterns[C/OL] //Proc of the 5th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2012). (2012-11-02)[2021-04-20]. https://hal.inria.fr/hal-00747841/document

[19] Yan Z. Weird new tricks for browser fingerprinting[EB/OL]. 2015[2020-10-12]. https://zyan.scripts.mit.edu/presentations/toorcon2015.pdf

[20] Smith M, Disselkoen C, Narayan S, et al. Browser history re: Visited[C/OL] //Proc of the 12th USENIX Workshop on Offensive Technologies (WOOT 18). 2018[2020-05-11]. https://www.usenix.org/system/files/conference/woot18/woot18-paper-smith.pdf

[21] Huang Anxin, Zhu Chen, Wu Deweb, et al. An adaptive method for cross-platform browser history sniffing[C/OL] //Proc of the 2nd Measurements, Attacks, and Defenses for the Web Workshop. Rosten: The Internet Society, 2020[2021-03-08]. https://www.ndss-symposium.org/wp-content/uploads/2020/02/23006.pdf

[22] Boda K, Földes Á M, Gulyás G G, et al. User tracking on the web via cross-browser fingerprinting[C] //Proc of the 16th Nordic Conf on Secure IT Systems. Berlin: Springer, 2011: 31−46

[23] Fifield D, Egelman S. Fingerprinting web users through font metrics[C] //Proc of the 19th Int Conf on Financial Cryptography and Data Security. Berlin: Springer, 2015: 107−124

[24] Saito T, Takahashi K, Yasuda K, et al. OS and application identification by installed fonts[C] //Proc of the 30th Int Conf on Advanced Information Networking and Applications (AINA). Piscataway NJ: IEEE, 2016: 684−689

[25] Kotowicz K, OSBORNAND K. Advanced chrome extension exploitation leveraging API powers for better evil [EB/OL]. Black Hat USA, 2012[2020-05-11]. https://paper.bobylive.com/Meeting_Papers/BlackHat/USA-2012/BH_US_12_Osborn_Kotowicz_Advanced_Chrome_Extension_WP.pdf

[26] Sjösten A, Van Acker S, Sabelfeld A. Discovering browser extensions via web accessible resources[C] //Proc of the 7th ACM Conf on Data and Application Security and Privacy. New York: ACM, 2017: 329−336

[27] Sjösten A, Van Acker S, Picazo-Sanchez P, et al. Latex gloves: Protecting browser extensions from probing and revelation attacks[EB/OL]. 2018[2020-05-11]. http://singularity.be/public/papers/latexgloves.pdf

[28] Sanchez-Rola I, Santos I, Balzarotti D. Extension breakdown: Security analysis of browsers extension resources control policies[C] //Proc of the 26th USENIX Security Symp. Berkeley, CA: USENIX Association, 2017: 679−694

[29] Starov O, Nikiforakis N. XHOUND: Quantifying the fingerprintability of browser extensions[C] //Proc of the 38th IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2017: 941−956

[30] Karami S, Ilia P, Solomos K, et al. Carnus: Exploring the privacy threats of browser extension fingerprinting[C/OL] //Proc of the 27th Network and Distributed System Security Symp (NDSS). Rosten: The Internet Society, 2020[2020-05-11]. https://www.ndss-symposium.org/wp-content/uploads/2020/02/24383-paper.pdf

[31] Laperdrix P, Starov O, Chen Quan, et al. Fingerprinting in Style: Detecting browser extensions via injected style sheets[C/OL] //Proc of the 30th USENIX Security Symp. Berkeley, CA: USENIX Association, 2021[2021-05-31]. https://www.usenix.org/system/files/sec21fall-laperdrix.pdf

[32] Nikiforakis N, Kapravelos A, Joosen W, et al. Cookieless monster: Exploring the ecosystem of Web-based device fingerprinting[C] // Proc of the 34th IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2013: 541−555

[33] Mulazzani M, Reschl P, Huber M, et al. Fast and reliable browser identification with Javascript engine fingerprinting[C] //Proc of the 7th Web 2.0 Workshop on Security and Privacy (W2SP). 2013: 4−14

[34] Takei N, Saito T, Takasu K, et al. Web browser fingerprinting using only cascading style sheets[C]//Proc of the 10th Int Conf on Broadband and Wireless Computing, Communication and Applications (BWCCA). Piscataway, NJ: IEEE, 2015: 57−63

[35] Schwarz M, Lackner F, Gruss D. JavaScript template attacks: Automatically inferring host information for targeted exploits[EB/OL]. 2019[2021-05-31]. https://www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_01B4_Schwarz_paper.pdf

[36] Mowery K, Bogenreif D, Yilek S, et al. Fingerprinting information in JavaScript implementations[C/OL] //Proc of the 5th Web 2.0 Workshop on Security and Privacy(W2SP). 2011[2021-04-20]. https://cseweb.ucsd.edu/~kmowery/papers/js-fingerprinting.pdf

[37] Solomos K, Kristoff J, Kanich C, et al. Persistent tracking in modern browsers[C/OL] //Proc of the 28th Symp on Network and Distributed System Security (NDSS). Rosten: The Internet Society, 2021[2021-05-31]. https://www.ndss-symposium.org/wp-content/uploads/ndss2021_1C-5_24202_paper.pdf

[38] Bansal C, Preibusch S, Milic-Frayling N. Cache timing attacks revisited: Efficient and repeatable browser history, OS and network sniffing[C] //Proc of the 30th Int Information Security and Privacy Conf. Berlin: Springer, 2015: 97−111

[39] Solís-Martínez, J, Espada J P, Crespo R G, et al. UXJs: Tracking and analyzing Web usage information with a Javascript oriented approach[J]. IEEE Access, 2020, 8: 43725−43735

[40] Navalpakkam V, Churchill E. Mouse tracking: Measuring and predicting users' experience of Web-based content[C] //Proc of the SIGCHI Conf on Human Factors in Computing Systems. New York: ACM, 2012: 2963−2972

[41] Mueller F, Lockerd A. Cheese: Tracking mouse movement activity on websites, a tool for user modeling[C/OL] //Proc of the 1st CHI Conf on Human Factors in Computing Systems. 2001[2021-04-20]. https://www.cc.gatech.edu/fac/athomaz/papers/cheese.pdf

[42] Katerina T, Nicolaos P, Charalampos Y. Mouse tracking for Web marketing: Enhancing user experience in Web application software by measuring self-efficacy and hesitation levels[J]. International Journal on Strategic Innovative Marketing, 2014(1): 233−247

[43] Lipp M, Gruss D, Schwarz M, et al. Practical keystroke timing attacks in sandboxed Javascript[C] //Proc of the 22nd European Symp on Research in Computer Security. Berlin: Springer, 2017: 191−209

[44] Mowery K, Shacham H. Pixel perfect: Fingerprinting canvas in HTML5[C/OL] //Proc of the 6th Web 2.0 Workshop on Security and Privacy(W2SP). 2012[2021-04-20]. https://cseweb.ucsd.edu/~kmowery/papers/html5-fingerprint.pdf

[45] Le H, Fallace F, Barlet-Ros P. Towards accurate detection of obfuscated web tracking[C/OL] //Proc of the 5th IEEE Int Workshop on Measurement and Networking (M&N). Piscataway, NJ: IEEE, 2017[2021-04-20]. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8078365

[46] Laperdrix P, Rudametkin W, Baudry B. Beauty and the beast: Diverting modern Web browsers to build unique browser fingerprints[C] //Proc of the 37th IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2016: 878−894

[47] Daud N I, Haron G R, Othman S S S. Adaptive authentication: Implementing random canvas fingerprinting as user attributes factor[C] //Proc of the 4th IEEE Symp on Computer Applications & Industrial Electronics (ISCAIE). Piscataway, NJ: IEEE, 2017: 152−156

[48] Raschke P, Küpper A. Uncovering canvas fingerprinting in real-time and analyzing its Usage for Web-tracking[C/OL]//Proc of the Workshops der INFORMATIK 2018-Architekturen, Prozesse, Sicherheit und Nachhaltigkeit. Köllen Druck+ Verlag GmbH. 2018[2021-04-20]. https://dl.gi.de/bitstream/handle/20.500.12116/17237/3032414_GI_P_285_09.pdf?sequence=1&isAllowed=y

[49] Englehardt S, Narayanan A. Online tracking: A 1-million-site measurement and analysis[C] //Proc of the 16th ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2016: 1388−1401

[50] Englehardt S, Eubank C, Zimmerman P, et al. OpenWPM: An automated platform for Web privacy measurement[J/OL]. Manuscript, 2015[2021-05-31]. https://senglehardt.com/papers/openwpm_03 − 2015.pdf

[51] Upathilake R, Li Yingkun, Matrawy A. A classification of Web browser fingerprinting techniques[C/OL] //Proc of the 7th Int Conf on New Technologies, Mobility and Security (NTMS). Piscataway, NJ: IEEE, 2015[2021-04-20]. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7266460

[52] Nakibly G, Shelef G, Yudilevich S. Hardware fingerprinting using HTML5[J]. arXiv, preprint arXiv: 1503.01408, 2015

[53] Cao Yinzhi, Song Li, Erik W. (Cross-) browser fingerprinting via OS and hardware level features[C/OL] //Proc of the 24th Symp on Network and Distributed System Security (NDSS). 2017[2021-05-31]. https://www.yinzhicao.org/TrackingFree/cross-browsertracking_NDSS17.pdf

[54] Saito T, Yasuda K, Ishikawa T, et al. Estimating CPU features by browser fingerprinting[C] //Proc of the 10th Int Conf on Innovative

Mobile and Internet Services in Ubiquitous Computing (IMIS). Piscataway, NJ: IEEE, 2016: 587−592

[55] Saito T, Yasuda K, Tanabe K, et al. Web browser tampering: Inspecting CPU features from side-channel information[C] //Proc of Int Conf on Broadband and Wireless Computing, Communication and Applications. Berlin: Springer, 2017: 392−403

[56] Diaz C, Olejnik L, Acar G, et al. The leaking battery: A privacy analysis of the HTML5 battery status API[G] //LNCS 9481: Int Workshop on Data Privacy Management. Berlin: Springer, 2015: 254−263

[57] Sanchez-Rola I, Santos I, Balzarotti D. Clock around the clock: Time-based device fingerprinting[C] //Proc of the 18th ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2018: 1502−1514

[58] Mishra V, Laperdrix P, Vastel A, et al. Don't count me out: On the relevance of IP address in the tracking ecosystem[C] //Proc of the Web Conf. New York: ACM, 2020: 808−815

[59] Hosoi R, Saito T, Ishikawa T, et al. A browser scanner: Collecting intranet information[C] //Proc of the 19th Int Conf on Network-Based Information Systems (NBiS). Piscataway, NJ: IEEE, 2016: 140−145

[60] Al-Fannah N M, Li Wanpeng. Not all browsers are created equal: Comparing Web browser fingerprintability[C] //Proc of the Int Workshop on Security. Berlin: Springer, 2017: 105−120

[61] Hazhirpasand M, Ghafari M. One leak is enough to expose them all[C] //Proc of the Int Symp on Engineering Secure Software and Systems. Berlin: Springer, 2018: 61−76

[62] Sy E, Burkert C, Federrath H, et al. Tracking users across the Web via TLS session resumption[C] //Proc of the 34th Annual Computer Security Applications Conf. New York: ACM, 2018: 289−299

[63] Jia Yaoqi, Dong Xinshu, Liang Zhenkai, et al. I know where you've been: Geo-inference attacks via the browser cache[J]. IEEE Internet Computing, 2014, 19(1): 44−53

[64] Klein A, Pinkas B. DNS cache-based user tracking[C/OL] //Proc of the 26th Symp on Network and Distributed System Security (NDSS). 2019[2021-05-31]. https://www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_04B-4_Klein_paper.pdf

[65] Mirheidari S A, Arshad S, Onarlioglu K, et al. Cached and confused: Web cache deception in the wild[C] //Proc of the 29th USENIX Security Symp. Berkeley, CA: USENIX Association, 2020: 665−682

[66] Solomos K, Ilia P, Ioannidis S, et al. Cross-device tracking: Systematic method to detect and measure CDT[J]. arXiv preprint, arXiv: 1812.11393, 2018

[67] Gómez-Boix A, Laperdrix P, Baudry B. Hiding in the crowd: An analysis of the effectiveness of browser fingerprinting at large scale[C] //Proc of the 2018 World Wide Web Conf. New York: ACM, 2018: 309−318

[68] Brookman J, Rouge P, Alva A, et al. Cross-device tracking: Measurement and disclosures[J]. Proceedings on Privacy Enhancing Technologies, 2017(2): 133−148

[69] Kane S K, Karlson A K, Meyers B R, et al. Exploring cross-device web use on PCs and mobile devices[C] //Proc of the IFIP Conf on Human-Computer Interaction. Berlin: Springer, 2009: 722−735

[70] Karakaya C, Toğuç H, Kuzu R S, et al. Survey of cross device matching approaches with a case study on a novel database[C] //Proc of the 3rd Int Conf on Computer Science and Engineering (UBMK). Piscataway, NJ: IEEE, 2018: 139−144

[71] Yen Tingfang, Xie Yinglian, Yu Fang, et al. Host fingerprinting and tracking on the Web: Privacy and security implications[C/OL] // Proc of the 19th Symp on Network and Distributed System Security(NDSS). Rosten: The Internet Society, 2012[2021-03-20]. https://www.ndss-symposium.org/wp-content/uploads/2017/09/11_3.pdf

[72] Zimmeck S, Li J S, Kim H, et al. A privacy analysis of cross-device tracking[C] //Proc of the 26th USENIX Security Symp. Berkeley, CA: USENIX Association, 2017: 1391−1408

[73] Díaz-Morales R. Cross-device tracking: Matching devices and cookies[C] //Proc of the IEEE Int Conf on Data Mining Workshop (ICDMW). Piscataway, NJ: IEEE, 2015: 1699−1704

[74] Li Song, Cao Yinzhi. Who touched my browser fingerprint? A large-scale measurement study and classification of fingerprint dynamics[C] //Proc of the 20th ACM Internet Measurement Conf. New York: ACM, 2020: 370−385

[75] Yamada T, Saito T, Takasu K, et al. Robust identification of browser fingerprint comparison using edit distance[C] //Proc of the 10th Int Conf on Broadband and Wireless Computing, Communication and Applications (BWCCA). Piscataway, NJ: IEEE, 2015: 107−113

[76] Liu Xiaofeng, Liu Qixu, Wang Xiaoxi, et al. Fingerprinting Web browser for tracing anonymous Web attackers[C] //Proc of the 1st IEEE Int Conf on Data Science in Cyberspace (DSC). Piscataway, NJ: IEEE, 2016: 222−229

[77] Jiang Wei, Wang Xiaoxi, Song Xinfang, et al. Tracking your browser with high-performance browser fingerprint recognition model[J]. China Communications, 2020, 17(3): 168−175

[78] Dong Shichuan, Farha F, Cui Shan, et al. CPG-FS: A CPU performance graph based device fingerprint scheme for devices identification and authentication[C] //Proc of the IEEE 38th Int Conf on Dependable, Autonomic and Secure Computing, 17th Int Conf on Pervasive Intelligence and Computing, 5th Int Conf on Cloud and Big Data Computing, 4th Int Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech). Piscataway, NJ: IEEE, 2019: 266−270

[79] Vastel A, Laperdrix P, Rudametkin W, et al. Fp-stalker: Tracking browser fingerprint evolutions[C] //Proc of the 39th IEEE Symp on Security and Privacy (SP). Piscataway, NJ: IEEE, 2018: 728−741

[80] Liu Qixu, Liu Xinyu, Luo Cheng, et al. Android browser fingerprinting identification method based on bidirectional recurrent neural network[J]. Journal of Computer Research and Development, 2020, 57(11): 2294−2311 (in Chinese)
(刘奇旭, 刘心宇, 罗成, 等. 基于双向循环神经网络的安卓浏览器指纹识别方法[J]. 计算机研究与发展, 2020, 57(11): 2294−2311)

[81] Tanabe K, Hosoya R, Saito T. Combining features in browser fingerprinting[C] //Proc of the 18th Int Conf on Broadband and Wireless Computing, Communication and Applications. Berlin:

Springer, 2018: 671−681

[82] Iqbal U, Englehardt S, Shafiq Z. Fingerprinting the fingerprinters: Learning to detect browser fingerprinting behaviors[C] //Proc of the 42nd IEEE Symp on Security and Privacy (SP). Piscataway, NJ: IEEE, 2021: 1143−1161

[83] Bird S, Mishra V, Englehardt S, et al. Actions speak louder than words: Semi-supervised learning for browser fingerprinting detection[J]. arXiv preprint, arXiv: 2003.04463, 2020

[84] Durey A, Laperdrix P, Rudametkin W, et al. An iterative technique to identify browser fingerprinting scripts[J]. arXiv preprint, arXiv: 2103.00590, 2021

[85] Acar G, Juarez M, Nikiforakis N, et al. FPDetective: Dusting the Web for fingerprinters[C] //Proc of the 13th ACM SIGSAC Conf on Computer & Communications Security. New York: ACM, 2013: 1129−1140

[86] Hannak A, Soeller G, Lazer D, et al. Measuring price discrimination and steering on e-commerce Web sites[C] //Proc of the 14th Conf on Internet Measurement. New York: ACM, 2014: 305−318

[87] Mathur A, Vitak J, Narayanan A, et al. Characterizing the use of browser-based blocking extensions to prevent online tracking[C] //Proc of the 14th Symp on Usable Privacy and Security (SOUPS 2018). Berkeley, CA: USENIX Association, 2018: 103−116

[88] Meng Wei, Lee B, Xing Xinyu, et al. TrackMeOrNot: Enabling flexible control on Web tracking[C] //Proc of the 25th Int Conf on World Wide Web. New York: ACM, 2016: 99−109

[89] Torres C F, Jonker H, Mauw S. FP-Block: Usable Web privacy by controlling browser fingerprinting[C] //Proc of the 20th European Symp on Research in Computer Security. Berlin: Springer, 2015: 3−19

[90] Al-Fannah N M, Li Wanpeng, Mitchell C J. Beyond cookie monster amnesia: Real world persistent online tracking[C] //Proc of the 33rd Conf on Information Security. Berlin: Springer, 2018: 481−501

[91] Starov O, Nikiforakis N. Extended tracking powers: Measuring the privacy diffusion enabled by browser extensions[C] //Proc of the 26th Int Conf on World Wide Web. New York: ACM, 2017: 1481−1490

[92] kkapsner. Canvas Blocker[EB/OL]. [2020-04-05]. https://github.com/k- kapsner/CanvasBlocker

[93] Obida M A, Obeidat S, Holst J, et al. Canvas Deceiver−A new defense mechanism against canvas fingerprinting[J/OL]. The Journal on Systemics, Cybernetics and Informatics. 2020[2021-05-31]. http://www.iiisci.org/journal/PDV/sci/pdfs/SA899XU20.pdf

[94] Leiva L A, Arapakis I, Iordanou C. My mouse, My Rules: Privacy issues of behavioral user profiling via mouse tracking[C] //Proc of the 6th ACM SIGIR Conf on Human Information Interaction and Retrieval. New York: ACM, 2021: 51−61

[95] Xiang Pan, Cao Yinzhi, Yan Chen. I do not know what you visited last summer: Protecting users from third-party Web tracking with trackingFree browser[C/OL] //Proc of the 22nd Annual Network and Distributed System Security Symp (NDSS). Rosten: The Internet Society, 2015[2021-05-31]. https://users.cs.northwestern.edu/~ychen/Papers/trackingfree_NDSS15.pdf

[96] Laperdrix P, Rudametkin W, Baudry B. Mitigating browser fingerprint tracking: Multi-level reconfiguration and diversification[C] //Proc of the 10th Int Symp on Software Engineering for Adaptive and Self-Managing Systems. Piscataway, NJ: IEEE, 2015: 98−108

[97] Baumann P, Katzenbeisser S, Stopczynski M, et al. Disguised chromium browser: Robust browser, flash and canvas fingerprinting protection[C] //Proc of the 15th ACM on Workshop on Privacy in the Electronic Society. New York: ACM, 2016: 37−46

[98] Yu Zhonghao, Macbeth S, Modi K, et al. Tracking the trackers[C] //Proc of the 25th Int Conf on World Wide Web. New York: ACM, 2016: 121−132

[99] Macbeth S. Tracking the trackers: Analysing the global tracking landscape with ghostrank. Technical report, Ghostery[EB/OL]. 2017[2021-04-20]. https://www.medienkraft.at/cms/wp-content/uploads/2018/10/user-tracking-studie-ghostery.pdf

[100] Starov O, Laperdrix P, Kapravelos A, et al. Unnecessarily identifiable: Quantifying the fingerprintability of browser extensions due to bloat[C] //Proc of the 28th World Wide Web. New York: ACM, 2019: 3244−3250

[101] FaizKhademi A, Zulkernine M, Weldemariam K. FPGuard: Detection and prevention of browser fingerprinting[C] //Proc of the 29th IFIP Annual Conf on Data and Applications Security and Privacy. Berlin: Springer, 2015: 293−308

[102] Nikiforakis N, Joosen W, LiVshits B. PriVaricator: Deceiving fingerprinters with little white lies[C] //Proc of the 24th Int Conf on World Wide Web. New York: ACM, 2015: 820−830

[103] Laperdrix P, Baudry B, Mishra V. FPRandom: Randomizing core browser objects to break advanced device fingerprinting techniques[C] //Proc of the 6th Int Symp on Engineering Secure Software and Systems. Berlin: Springer, 2017: 97−114

[104] Yokoyama S, Uda R. A proposal of preventive measure of pursuit using a browser fingerprint[C/OL] //Proc of the 9th Int Conf on Ubiquitous Information Management and Communication. New York: ACM, 2015[2021-04-20]. https://dl.acm.org/doi/pdf/10.1145/2701126.2701210

[105] Wu Shujiang, Li Song, Cao Yinzhi, et al. Rendered private: Making {GLSL} execution uniform to prevent WebGL-based browser fingerprinting[C] //Proc of the 28th USENIX Security Symp. Berkeley, CA: USENIX Association, 2019: 1645−1660

[106] Jueckstock J, Kapravelos A. VisibleV8: In-browser monitoring of JavaScript in the wild[C] //Proc of the 19th Internet Measurement Conf. New York: ACM, 2019: 393−405

[107] Andriamilanto N, Allard T, Le Guelvouit G. FPSelect: Low-cost browser fingerprints for mitigating dictionary attacks against Web authentication mechanisms[C] //Proc of the Annual Computer Security Applications Conf. New York: ACM, 2020: 627−642

[108] Andriamilanto N, Allard T. BrFAST: A tool to select browser fingerprinting attributes for Web authentication according to a usability-security trade-off[C] //Proc of the Web Conf. New York: ACM, 2021: 701−704

[109] Solomos K, Ilia P, Ioannidis S, et al. Automated measurements of

cross-device tracking[C] //Proc of the 13th Int Workshop on Information and Operational Technology Security Systems. Berlin: Springer, 2018: 73−80

[110] Solomos K, Ilia P, Ioannidis S, et al. {TALON}: An automated framework for cross-device tracking detection[C] //Proc of the 22nd Int Symp on Research in Attacks, Intrusions and Defenses (RAID 2019). Berlin: Springer, 2019: 227−241.

[111] Trickel E, Starov O, Kapravelos A, et al. Everyone is different: Client-side diversification for defending against extension fingerprinting[C] //Proc of the 28th USENIX Security Symp. Berkeley, CA: USENIX Association, 2019: 1679−1696

[112] Gómez-Boix A, Frey D, Bromberg Y D, et al. A collaborative strategy for mitigating tracking through browser fingerprinting[C] //Proc of the 6th ACM Workshop on Moving Target Defense. New York: ACM, 2019: 67−78

[113] Overdorf R, Juarez M, Acar G, et al. How unique is your onion? An analysis of the fingerprintability of Tor onion services[C] //Proc of the 17th ACM SIGSAC Conf on Computer and Communications Security. New York: ACM. 2017: 2021−2036

[114] Datta A, Lu Jianan, Tschantz M C. The effectiveness of privacy enhancing technologies against fingerprinting[J]. arXiv preprint, arXiv: 1812.03920, 2018

[115] Lanze F, Panchenko A, Engel T. A formalization of fingerprinting techniques[C] //Proc of the 2015 IEEE 14th Trustcom/1st BigDataSE/13th ISPA. Piscataway, NJ: IEEE, 2015: 818−825

[116] Vastel A. Tracking versus security: Investigating the two facets of browser fingerprinting[D/OL]. 2019[2021-05-20]. https://tel.archives-ouvertes.fr/tel-02343930/do- Cument

[117] Antonio E, Fajardo A, Medina R. Tracking browser fingerprint using rule based algorithm[C] //Proc of the 16th IEEE Int Colloquium on Signal Processing & Its Applications. Piscataway, NJ: IEEE, 2020: 225−229

[118] Vastel A, Rudametkin W, Rouvoy R. FP-TESTER: Automated testing of browser fingerprint resilience[C] //Proc of the 3rd IEEE European Symp on Security and Privacy Workshops. Piscataway, NJ: IEEE, 2018: 103−107

[119] Vastel A, Laperdrix P, Rudametkin W, et al. FP-scanner: The privacy implications of browser fingerprint inconsistencies[C] //Proc of the 27th USENIX Security Symp. Berkeley, CA: USENIX Association, 2018: 135−150

[120] Queiroz J S, Feitosa E L. A Web browser fingerprinting method based on the Web audio API[J]. The Computer Journal, 2019, 62(8): 1106−1120

[121] Gulyas G, Some D F, Bielova N, et al. To extend or not to extend: On the uniqueness of browser extensions and Web logins[C] //Proc of the 17th Workshop on Privacy in the Electronic Society. New York: ACM, 2018: 14−27

[122] Abouollo A, Almuhammadi S. Detecting malicious user accounts using canvas fingerprint[C] //Proc of the 8th Int Conf on Information and Communication Systems. Piscataway, NJ: IEEE, 2017: 358−361

[123] Unger T, Mulazzani M, Frühwirt D, et al. Shpf: Enhancing http (s) session security with browser fingerprinting[C] //Proc of the 8th Int Conf on Availability, Reliability and Security. Piscataway, NJ: IEEE, 2013: 255−261

[124] Jonker H, Krumnow B, Vlot G. Fingerprint surface-based detection of Web bot detectors[C] //Proc of the 24th European Symp on Research in Computer Security. Berlin: Springer, 2019: 586−605

[125] Vastel A, Rudametkin W, Rouvoy R, et al. FP-Crawlers: Studying the resilience of browser fingerprinting to block crawlers[C/OL] //Proc of the NDSS Workshop on Measurements, Attacks, and Defenses for the Web. 2020[2021-04-20]. https://www.ndss-symposium.org/wp-content/uploads/2020/02/23010.pdf

[126] Agarwal V, Vekaria Y, Agarwal P, et al. Under the spotlight: Web tracking in Indian partisan news websites[J]. arXiv preprint, arXiv: 2102.03656, 2021

[127] Takahashi T, Kruegel C, Vigna G, et al. Tracing and analyzing Web access paths based on user-side data collection: How do users reach malicious URLs?[C] //Proc of the 23rd Int Symp on Research in Attacks, Intrusions and Defenses (RAID 2020). Berlin: Springer, 2020: 93−106

[128] Jia, Zhaopeng, Cui Xiang, Liu Qixu, et al. Micro-honeypot: Using browser fingerprinting to track attackers[C] //Proc of the 3rd IEEE Int Conf on Data Science in Cyberspace (DSC). Piscataway, NJ: IEEE, 2018: 197−204

[129] Li Bo, Vadrevu P, Lee K H, et al. JSgraph: Enabling reconstruction of Web attacks via efficient tracking of live in-browser Javascript executions[C/OL] //Proc of the 25th Network and Discributed System Security Symp (NDSS). 2018 [2020-05-11]. https://web.archive.org/web/20180307204133id_/http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_07B-4_Li_paper.pdf

[130] Oh J, Lee S, Lee S. Advanced evidence collection and analysis of web browser activity[J/OL]. Digital Investigation, 2011[2021-04-20]. https://dl.acm.org/doi/abs/10.1016/j.diin.2011.05.008

[131] Winter P, Edmundson A, Roberts L M, et al. How do Tor users interact with onion services?[C] //Proc of the 27th USENIX Security Symp. Berkeley, CA: USENIX Association, 2018: 411−428

[132] Fiore U, Castiglione A, De Santis A, et al. Countering browser fingerprinting techniques: Constructing a fake profile with Google Chrome[C] //Proc of the 17th Int Conf on Network-Based Information Systems. Piscataway, NJ: IEEE, 2014: 355−360

[133] Luangmaneerote S, Zaluska E, Carr L. Survey of existing fingerprint countermeasures[C] //Proc of the 2016 Int Conf on Information Society. Piscataway, NJ: IEEE, 2016: 137−141

[134] Samarasinghe N, Mannan M. Towards a global perspective on Web tracking[J]. Computers & Security, 2019(87): 101569

[135] Luangmaneerote S, Zaluska E, Carr L. Inhibiting browser fingerprinting and tracking[C] //Proc of the IEEE 3rd Int Conf on Big Data Security on Cloud (BigDataSecurity), IEEE Int Conf on High Performance and Smart Computing (HPSC), and IEEE Int Conf on Intelligent Data and Security (IDS). Piscataway, NJ: IEEE, 2017: 63−68



Wang Xiaoxi, born in 1990. PhD candidate. Her main research interests include cyber security and Web tracking.

王晓茜, 1990 年生. 博士研究生. 主要研究方向为网络安全、Web 追踪.



Liu Qixu, born in 1984. PhD, professor, PhD supervisor. His main research interests include Web security and attribution and forensic.

刘奇旭, 1984 年生. 博士, 教授, 博士生导师. 主要研究方向为 Web 安全和溯源取证.



Liu Chaoge, born in 1986. PhD, associate professor. His main research interests include malware and Web security.

刘潮歌, 1986 年生. 博士, 副教授. 主要研究方向为恶意代码和 Web 安全.



Zhang Fangjiao, born in 1989, PhD candidate. Her main research interests include cyber attack, defense and cybersecurity talents evaluation.

张方娇, 1989 年生. 博士研究生. 主要研究方向为网络攻防和网安人才评估.



Liu Xinyu, born in 1997. PhD candidate. Her main research interests include Web security and Android security.

刘心宇, 1997 年生. 博士研究生. 主要研究方向为 Web 安全和 Android 安全.



Cui Xiang, born in 1978. PhD, professor, PhD supervisor. His main research interests include malware analysis and Web security.

崔　翔, 1978 年生. 博士, 教授, 博士生导师. 主要研究方向为恶意代码分析和 Web 安全.