Evaluate prevalence of browser fingerprinting in advertising and the effectiveness of existing defences

CHAN KING SEN

Submitted as part of the requirements for the award of the
MSc in Information Security
at Royal Holloway University of London.

Information Security Group
Royal Holloway, University of London
September 2025

I declare that parts of this submission have contributions from AI software and that it aligns with acceptable use as specified as part of the assignment brief/guidance and is consistent with good academic practice. The content can still be considered as my own words. I understand that as long as my use falls within the scope of appropriate use as defined in the assessment brief/guidance then this declaration will not have any direct impact on the grades awarded. Below are details of the generative AI tool(s) and how I have used them.

I acknowledge use of Copilot, https://copilot.microsoft.com, with the prompts available in Appendix D.1 *to perform:*

i. *English Grammar and Spell Check*

Acknowledgements

I would like to express my sincere gratitude to my supervisor, who has provided invaluable support and guidance throughout the course of my dissertation. Without her assistance, I would not have been able to complete this work.

I am also deeply thankful to my lovely wife, whose unwavering support and countless sacrifices have made this journey possible. I am truly grateful for everything she has done to support my studies.

Finally, I would like to thank everyone who contributed in any way to the completion of this dissertation, including my fellow course mates, personal advisor, module leads and teaching assistants.

Table of Contents

Acknowledgements 3

List of Figures 7

List of Tables 8

List of Abbreviations & Acronyms 9

Abstract 10

1 Introduction 11

1.1 Motivation 11

1.2 Contribution 12

1.3 Objectives 13

1.4 Scope 13

1.5 Document Structure 14

2 Background 16

2.1 Browser Fingerprinting - Primer 16

2.2 Applications of Browser Fingerprinting 18

2.2.1 Web Tracking 18

2.2.2 Web Security 19

2.3 Browser Fingerprinting Methods 20

2.3.1 Audio Fingerprinting 22

2.3.2 Canvas fingerprinting 22

2.3.3 Font Fingerprinting 23

2.3.4 WebGL fingerprinting 24

2.3.5 WebRTC fingerprinting 24

2.4 Browser Fingerprinting Countermeasures 25

2.4.1 Randomisation 25

2.4.2 Normalisation 26

2.4.3 Heuristic 26

3 Related Work 28

3.1 Origins of browser fingerprinting 28

3.2 Prevalence and Scope of browser fingerprinting 28

3.3 Browser fingerprinting Countermeasures 30

3.4 Summary 31

4 Design Methodology 32

4.1 Ethical Consideration 33

4.2 Web Measurement Study using Instrumentation Tool OpenWPM 34

4.2.1 Extending Capabilities and Configuration Settings of OpenWPM 34

4.2.2 Filtering Successful Visits and Categorisation of Websites 35
4.3 Identifying and Contextualising Fingerprinting Websites 35
4.3.1 Disconnect and Ghostery Tracking Protection List (TPL) 35
4.3.2 Normalising and Assigning Category Tags 37
4.3.3 Resolving Category Conflict 38
4.4 FPAnalyze Heuristic Framework 38
4.4.1 Technical Definitions of Invasive and General Fingerprinting 38
4.4.2 Individual Attribute-Based Heuristic Scoring and Threshold 39
4.4.3 Joint-Attributes Heuristic Scoring 40
4.5 Testing Methodology: Effectiveness of Chrome Browser Extension 41
4.5.1 Selection and Configuration of Browser Selection 41
4.5.2 Selection and Configuration of Browser Extensions 41
4.5.3 Selection of Test Cases 42
4.5.4 Criteria for Evaluation 43
4.6 Test Environment Setup 43
5 Implementation 44
5.1 Web Measurement Study Using Instrumentation Tool OpenWPM 44
5.1.1 Accepting All Cookies 44
5.1.2 Profiling of Top 10,000 Tranco Websites 45
5.1.3 Website Categorisation 46
5.2 Identifying and Contextualising Fingerprinting Websites 46
5.3 Implementation of FPAnalyze Heuristic Framework and Analysis 47
5.4 Testing Browser Extensions 47
5.4.1 Setup of Chrome Browser Profiles 48
5.4.2 Benchmark Test 48
5.4.3 Differentiating between Direct and Indirect Blocking from Network Logs 49
6 Results 50
6.1 Overview of Prevalence of Browser Fingerprinting 50
6.2 Prevalence of Browser Fingerprinting in Digital Advertising 52
6.3 Browser Fingerprinting Detection Performance of FPAnalyze 53
6.4 Attribute Collection Patterns in Browser Fingerprinting 54
6.5 Evaluation of Browser Extension Defence 55
7 Discussion 57
7.1 Adoption of Browser Fingerprinting associated with Google Advertising Platforms 57
7.2 Relationship between Browser Fingerprinting and Site Categories 57

7.3 Prevalence of Browser Fingerprinting for User Identification in Digital Advertising 58
7.4 Extent of Fingerprinting Attributes 58
7.5 Browser Extension as a Defence Mechanism 59
7.6 Limitations 59
7.7 Future Works 60
8 Conclusion 61
9 Bibliography 63
Appendies 67
Appendix A.1 67
Appendix A.2 67
Appendix B.1 67
Appendix C.1 68
Appendix D.1 68

List of Figures

Figure 1: Diagram showing how device fingerprints are collected [21] 16
Figure 2: Diagram showing Role of Browser Fingerprinting in Online Advertising Network [23] 17
Figure 3: Diagram showing the set of attributes used to generate a browser fingerprint [35] 20
Figure 4: Diagram describing the process of creating an audio fingerprint using AudioContext 22
Figure 5: Example of a canvas fingerprint generated from canvas fingerprinting code of FingerprintJS 23
Figure 6: Diagram demonstrating the pseudo code for canvas font fingerprinting [4] 24
Figure 7: Code snippet on how to get IP information in the browser [42] 25
Figure 8: Overview of Experimental Design Architecture 32
Figure 9: Classification of trackers used by Ghostery 37
Figure 10: Diagram of Attributes used in Attribute-Based Scoring Heuristics 40
Figure 11: Joint-Attributes Decision Threshold Algorithm 41
Figure 12: Sample Output of Manual Analysis 43
Figure 13: Examples of Cookie Banners to Accept 45
Figure 14: Diagram of Distributed Database Architecture 45
Figure 15: Snapshot of how Site Category is obtained from Norton Safe Web 46
Figure 16: Sample of a Translated FilterList Rule Set for Disconnect 47
Figure 17: Snapshot of Cache Architecture For Individual Tasks 47
Figure 18: Distribution of top ten web site categorisation according to Norton Safe Web 50
Figure 19: Distribution of top 10 Site Categories with Browser Fingerprinting 51
Figure 20: Distribution of Browser Fingerprinting Normalised Against Category Size 51
Figure 21: Charts of No. of Third-Pary Providers found on each Detected Website 52
Figure 22: Comparative Analysis of Feature Detection Statistics 54
Figure 23: Distribution of Cluster of General Fingerprinting Attributes 55
Figure 24: Distribution of Cluster of Invasive Fingerprinting Attributes 55
Figure 25:Comparison of Mitigation Rate across Range of Joint-Attributes Heuristic Score 56
Figure 26: Code snippet of integration of Fingerprinting with GTM and GPT. 57
Figure 27: JS Instrumentation Configuration as specified in Section 4.2.1 67

List of Tables

Table 1: Classification of Browser Attributes Based on their Entropy and Stability [20, 31, 36, 37] 21
Table 2: OpenWPM configuration used for Profiling 35
Table 3: Table of Disconnect Tracker Categories and Description 36
Table 4: Overview of Category Tags Classification 38
Table 5: Classification of Attributes in Attribute-Based Heuristics Scoring 39
Table 6: Joint-Attributes Heuristic Scoring Threshold 40
Table 7: Selection Ratings for the Five Selected Browser Extensions 42
Table 8: Detailed Setup Configuration of VM 44
Table 9: Selenium and Chrome Configuration 48
Table 10: Configuration Settings for Browser Extensions 48
Table 11: Distribution of Browser Extension Test Cases 48
Table 12: Chrome's Network Logs Showing Script Blocked by Browser Extension 49
Table 13: Sample Test Result Output 49
Table 14: Top 10 third parties performing Browser Fingerprinting 52
Table 15: Proportion of Tracker Categories associated with Browser Fingerprinting 53
Table 16: Heatmap of Normalised Attributes used for General Fingerprinting 54
Table 17: Heatmap of Normalised Attributes used for Invasive Fingerprinting 55
Table 18: Comparison of Performance of Browser Extensions 56
Table 19: Detailed versions of resources as specified in Section 5.2 67
Table 20: Tranco Top 14K OpenWPM Crawl Status specified in Section 6.1 67
Table 21: Distribution of Network Errors Encountered as specified in Section 6.1 67
Table 22: Detailed Test Cases Websites specified in Section 5.4.2 68

List of Abbreviations & Acronyms

AdTech	Advertising Technology
ads	advertisements
GDPR	General Data Protection Regulation 2016
CCPA	California Consumer Privacy Act 2018
API	Application Programming Interface
ETP	Enhanced Tracking Protection
ITP	Intelligent Tracking Prevention
OS	Operating System
JS	JavaScript
CDN	Content Delivery Networks
ML	Machine Learning
MV3	Manifest file Version 3
CTR	Click-Through Rate
GPU	Graphics Processing Unit
WebGL	Web Graphics Library
WebRTC	Web-based Real-Time Communication
NAT	Network Address Translation
ICE	Interactive Connectivity Establishment
MIME	Multipurpose Internet Mail Extensions
FHD	Full High Definition
CAPTCHAs	Completely Automated Public Turing tests to tell Computers and Humans Apart
DOM	Document Object Model
TPL	Tracking Protection Lists
VPN	Virtual Private Network
VM	Virtual Machine
CSV	Commas Separated Values
D+G	Disconnect and Ghostery

Abstract

User identification has a critical role in digital advertising, a multibillion-dollar global industry. With growing user privacy concerns and users increasingly blocking third-party cookies, the digital advertising industry has seemingly shifted to alternative methods for user tracking. Browser fingerprinting, a stateless technique opaque to users, has emerged as a potential candidate. This is especially so following Google's recent privacy policies changes permitting its use. To assess this trend, this study presents a large-scale measurement of the prevalence of browser fingerprinting across the top 10,000 most visited websites from the Tranco list, utilising the instrumentation tool OpenWPM and tracker classifications from Ghostery and Disconnect. Results show that 38.55% of websites employ browser fingerprinting, marking a 5% increase from the previous study conducted in 2023. Despite this growing prevalence, only 606 of the profiled websites were observed to employ invasive fingerprinting for user identification, with 388 identified for the purpose of digital advertising. The study also reaffirmed that News website publishers are more likely to engage in browser fingerprinting than publishers in other site categories. To enhance detection and analysis of browser fingerprinting at the attributes level, a dynamic analysis-based heuristic framework named FPAnalyze is introduced and applied to the measurement dataset collected. FPAnalyze demonstrated relatively good performance, identifying 1,240 websites employing invasive fingerprinting for user identification, approximately twice the prior detection rate. Nevertheless, both results indicated that browser fingerprinting has not entirely replaced tracking cookies as a key method for user identification. Additionally, five widely used browser extensions have been evaluated for their effectiveness in mitigating browser fingerprinting for user identification. While the overall effectiveness is found to be moderate, Ghostery and uBlock Origin Lite emerged as the best performing browser extensions, successfully blocking 58% of 40 targeted test cases. However, given the relatively low prevalence of invasive fingerprinting for user identification in advertising purposes, browser extensions can still be considered to offer an adequate layer of privacy protection for users.

1 Introduction

This chapter establishes an overview of the background of the dissertation subject on browser fingerprinting for user identification, outlining the research motivation, objectives, scope, and document structure for the remainder of this paper.

1.1 Motivation

Digital advertising is a multibillion-dollar global industry, with spending forecasted to reach USD$694 billion in 2024 [1]. It has enabled content creators to monetise their websites, generating revenue for publishing their content on the Internet for free. At the same time, it has also fuelled the growth of AdTech platforms and companies in a complex ecosystem to effectively deliver personalised advertisements to potential customers.

User identification is important in AdTech to ensure the effective display of ads to the target audience. Different processes and methods have been used by AdTech to identify and track users across various web sites. The primary method used is the creation of third-party web cookies to store stateful session information of the user's activities in the browser [2]. These cookies enabled advertisers to track a user's behaviour across multiple sites, thereby gaining insights to the user's interest to enable the delivery of highly targeted and personalised advertisements [3].

However, due to growing privacy concerns and stricter regulations such as GDPR 2016 and CCPA 2018, explicit user consent is required before the usage of these cookies for tracking and advertising could be stored [4]. In recent years, major web browser vendors such as Safari and Firefox had responded and moved in favour towards a "cookieless browsing" [5] model, where third-party cookies are restricted or blocked by default to enhance user privacy. Firefox implemented Enhanced Tracking Protection (ETP) in 2019, while Safari introduced Intelligent Tracking Prevention (ITP) in 2020. More recently in Q3 2024, Privacy Sandbox technology APIs has also been introduced in Chrome, the most popular browser with a global market share of 66.3% [6], to restrict cross-site tracking third-party cookies [7]. Furthermore, there is also a surge in the installation of third-party add-ons with privacy protection features on browsers for additional protection. In 2018, Newman [8] conducted a study to evaluate the effectiveness of five browser add-ons' capability in blocking tracking cookies. The findings revealed that they were able to largely reduce their creation and transmission by approximately 60 – 80%.

In response to the reduced effectiveness of cookies for identification and tracking across websites, AdTech companies have long started exploring alternative techniques, such as pixel tracking and browser fingerprinting, to complement cookie tracking [9, 10]. Between 2013 and 2024, multiple measurement studies had been conducted by academia on the prevalence of browser fingerprinting in the real-world. Although there were variations in the detection techniques used across these studies, their findings indicated an overall rising trend in browser fingerprinting adoption during this period [11]. In

addition, following Google's recent announcement in December 2024 regarding the relaxation of its privacy policy, advertisers using its advertising products are now permitted to employ browser fingerprinting techniques with effect from 16 February 2025 [12].

The recent introduction of privacy-enhancement technology in the Chrome browser, coupled with relaxation of Google's privacy policy, might drive AdTech companies to further adopt browser fingerprinting for user identification. Exacerbating the situation, Google's impending upgrade to Manifest File Version 3 (MV3) for Chrome browser extensions since 2020 has now come into force in 2025. The use of extensions built on MV2 is now disabled by default [13]. The MV3 update had raised concerns among privacy-concerned users and advertising blocker providers that the effectiveness of advertising blockers might be reduced following the shift towards the provision of a more restrictive API for detecting and blocking tracking domains. Lukic [14] had conducted a comparative analysis of four widely used advertising blockers, evaluating the effectiveness of their ad-blocking and anti-tracking capabilities across their MV3 and MV2 implementation. Although the findings indicated that the ad-blocking effectiveness remained consistent, the study's scope was restricted to the evaluation of the blocking of advertisement display across a selected set of websites.

Building on Ukani's 2023 insight that "Once Chrome deprecates third-party cookies, a measurement study could evaluate whether advertisers completely switch to tracking users via fingerprinting" [11], and considering the significant evolution of the digital landscape since then, it is both relevant and timely to revisit the measurement study to address the following research questions:

RQ1. Has browser fingerprinting become more prevalent, particularly, in digital advertising?

RQ2. What are the different sets of browser attributes that are collected by real-world websites or third-party services during browser fingerprinting for user identification?

RQ3. To what extent are current defence mechanisms effective in mitigating browser fingerprinting assuming a higher level of adoption of browser fingerprinting for user identification?

1.2 Contribution

The main contributions of this dissertation are as follows:

a. A large-scale measurement study of the prevalence of browser fingerprinting practices across the Tranco Top 10,000 websites, with a particular focus on its role in user identification within the digital advertising ecosystem.

b. Development of FPAnalyze, a dynamic analysis - based heuristic framework designed to detect browser fingerprinting, including the

spectrum of invasive and general fingerprinting techniques deployed in JavaScript code

c. Evaluating effectiveness of browser extensions in mitigating the risks posed by invasive browser fingerprinting aimed at user identification.

1.3 Objectives

This dissertation's primary objective is to examine the extent to which browser fingerprinting has been adopted as a method for user identification within the context of digital advertising. Previous studies have primarily focused on the prevalence of browser fingerprinting for both online advertising and tracking in a broad context [15, 16]. Apart from Iqbal et al.'s [4] briefly investigation into the advertising-related links between fingerprinting and cookie syncing vendors, there remains a noticeable gap in existing literatures.

Specifically, this dissertation aimed to:

a. Systematically detect instances of browser fingerprinting for user identification in different categories of websites on the surface web.

b. Identify category of websites most likely to deploy browser fingerprinting for user identification techniques.

c. Categorise the context in which they are employed and determine the extent to which browser fingerprinting is deployed specifically for user identification in digital advertising.

d. Examine the range of browser fingerprinting attributes collected and utilised in constructing a user's pseudo-identity, using a custom heuristic framework. This investigation seeks to inform users about the kind and amount of information collected when opting in to tracking or personalised advertising.

e. Evaluate the effectiveness of the existing browser extension defence mechanisms in safeguarding user privacy against browser fingerprinting for user identification.

1.4 Scope

The scope of this study would be restricted in lieu of the constraints and size of the subject matter. To ensure clarity and consistency throughout the remainder of the paper, and in light of the absence of an universally accepted definition for browser fingerprinting [4], this research adopts the following definitions with further technical details in Section 4.4.1:

1. Invasive Fingerprinting – Techniques that leverages an API to extract distinguishing information from a user's device that was not originally designed to do so [17].

2. General fingerprinting – Techniques that leverages browser features or properties that could be used directly or indirectly to track users [17].

3. Browser fingerprinting – includes the use of either invasive or general fingerprinting, or both.

4. Browser fingerprinting for user identification – Refers to the use of at least one invasive technique, combined with other general fingerprinting techniques.

In addition, given the limited amount of time and resources, this study's scope would be limited to:

1. Profile only the landing page of the top 10,000 most visited websites on the surface web using desktop browsers. Mobile sites are not included in scope. This approach is intended to ensure that the sample population remained representative of the broader web usage trends and manageable for in-depth analysis.

2. Categorise contexts of identified browser fingerprinting sites using curated and open-sourced Tracking Protection Lists (TPL)[1] provided by commercial vendors as the basis for contextual classification. This approach ensures standardisation and consistency in categorising tracking activities.

3. Categorise and organise the set of fingerprint attributes into distinct categories that reflect their native properties and functional roles. This approach aimed to allow users to grasp the breadth of the fingerprinting techniques without needing in-depth technical details.

4. Evaluate the effectiveness of only 5 popular commercial ad blockers in mitigating browser fingerprinting for user identification. All selected extensions must be MV3 compatible and installed via the Chrome extension store.

5. Each extension will be assessed in its default configuration to reflect typical user behaviour and provide an objective analysis of their out-of-the-box anti-browser fingerprinting capability.

1.5 Document Structure

The rest of the dissertation paper is organised as follows:

Section 2 provides the theoretical background for the study, exploring the role of browser fingerprinting as a user identifier within the digital advertising ecosystem (Section 2.1) and discusses its other broader applications (Section 2.2) such as in Web Security. This is followed by the exploration of the top five most widely adopted fingerprinting method that are highly stable and unique (Section 2.3), and a review of the existing countermeasures (Section 2.4).

Section 3 summarises the review of related literature on the origins of browser fingerprinting (Section 3.1), surveys earlier measurement studies that examined

[1] List of known websites or domains blocked or restricted by software to prevent tracking and monitoring of users' online activities

the prevalence of browser fingerprinting across the web and the evolution of detection techniques (Section 3.2). Finally, prior research focusing on various development and evaluation of browser fingerprinting countermeasures are discussed (Section 3.3).

Section 4 outlines the design methodology and ethical considerations (Section 4.1) for the measurement study (Section 4.2), the strategy and approach to identify and contextualise fingerprinting websites (Section 4.3). The heuristic-based FPAnalyze framework and its design rationale is also introduced to explore how it facilitates detection of invasive and general fingerprinting scripts (Section 4.4). The section ends with the discussion on the test methodology and environment setup used to evaluate the effectiveness of browser extensions in defending against browser fingerprinting (Section 4.5).

Section 5 describes the implementation details of the study, including the customisation and configuration of the instrumentation tool OpenWPM (Section 5.1), the steps and procedures to integrate Disconnect and Ghostery Tracking Protection Lists resources to identify and contextualise fingerprinting websites (Section 5.2) and the tools used to implement FPAnalyze (Section 5.3). The detail setup and configuration of the test environment to ensure a fair and objective assessment is further discussed (Section 5.4).

Section 6 presents the results on the prevalence of browser fingerprinting (Section 6.1), particularly in digital advertising (Section 6.2). The detection performance of FPAnalyze (Section 6.3) and observations of attribute collection trends (Section 6.4) are also discussed, followed by the evaluation results of the five shortlisted browser extensions (Section 6.5).

Section 7 begins with the analysis of the adoption of browser fingerprinting associated with Google's advertising services (Section 7.1), followed by an exploration of the relationship between fingerprinting and website categories (Section 7.2). An assessment of the level of threat posed by invasive fingerprinting (Section 7.3) and examination of the spectrum of attributes collected is also provided (Section 7.4), before offering an objective assessment if current browser extensions does provide sufficient privacy protection (Section 7.5). The section ends with the exploration of the limitations of current research (Section 7.6) and discussion of future works (Section 7.7).

Section 8 summarises the key outcomes and contributions of the study.

2 Background

This section outlines the essential theoretical background for four key areas of this project: (a) introduction to browser fingerprinting and its role in accurately identifying users in digital advertising, (b) overview of other applications of browser fingerprinting, (c) introduction of invasive fingerprinting methods and (d) review of browser fingerprinting countermeasures.

2.1 Browser Fingerprinting - Primer

Browser fingerprinting is one of the methods of identifying and tracking users online. It is commonly referred to as the process of either the direct or indirect collection of different browser characteristic information on a device that remain largely unchanged over time, and combining these attributes to create a unique fingerprint of a device [18]. Typical data collected includes screen resolution, installed fonts and plugins. While these attributes might lack uniqueness or stability individually, a combination of these attributes often results in an identifiable fingerprint of a particular device or user over an extended period [19, 20]. Figure 1 [21] describes the types of browser and system information collected, and how it is extracted from the browser to generate a fingerprint for a device.

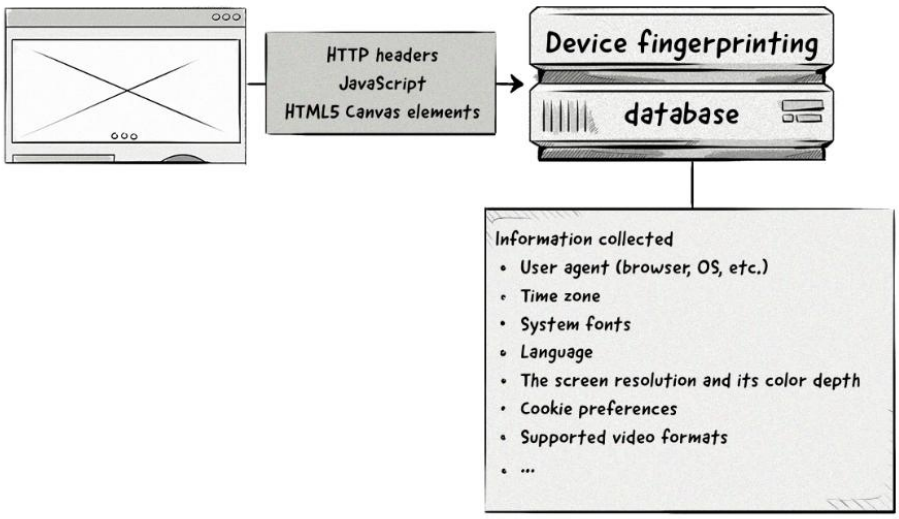

Figure 1: Diagram showing how device fingerprints are collected [21]

Businesses, such as news agencies, that publishes free content on their online websites are commonly known as web publishers. They monetise their websites by providing advertising space to allow third-party advertising brokers to advertise products on their page for a fee. Under the Pay-per-Click (PPC) model, the web publisher earns a portion of the advertising revenue when its visitors click on the advertisements [22]. In addition to serving advertisements, some advertising brokers might embed a fingerprinting script to collect a set of device-specific information silently in the background, unknown to the user. Augmented with contextual information such as the user's IP address and geolocation, this dataset is transmitted back to the broker. The broker might subsequently relay this information to its fingerprint provider to generate a unique identifier. Alternatively, the identifier could also be computed locally on

the user's device. The advertising broker can then correlate the generated identifier with their existing user profiles in their tracking database to identify new or returning users [23]. Behavioural metadata such as browsing patterns and click activities can also be incorporated to enrich these user profiles [16], thereby enabling the advertising brokers to not only monitor and analyse the user's activities over an extended time, but also across their affiliated websites. Leveraging the inferred preferences and interests from the user profiles collected, the brokers can algorithmically select and dynamically deliver personalised advertisements that are more appealing to the target profiles [8, 18]. This targeted strategy ultimately improves advertising efficacy by increasing the Click-Through Rate (CTR), leading to a higher advertisement revenue for both the website publisher and the advertisement brokers. Figure 2 [23] summarises the role of browser fingerprinting and the flow of information between the different entities in an online advertising network.

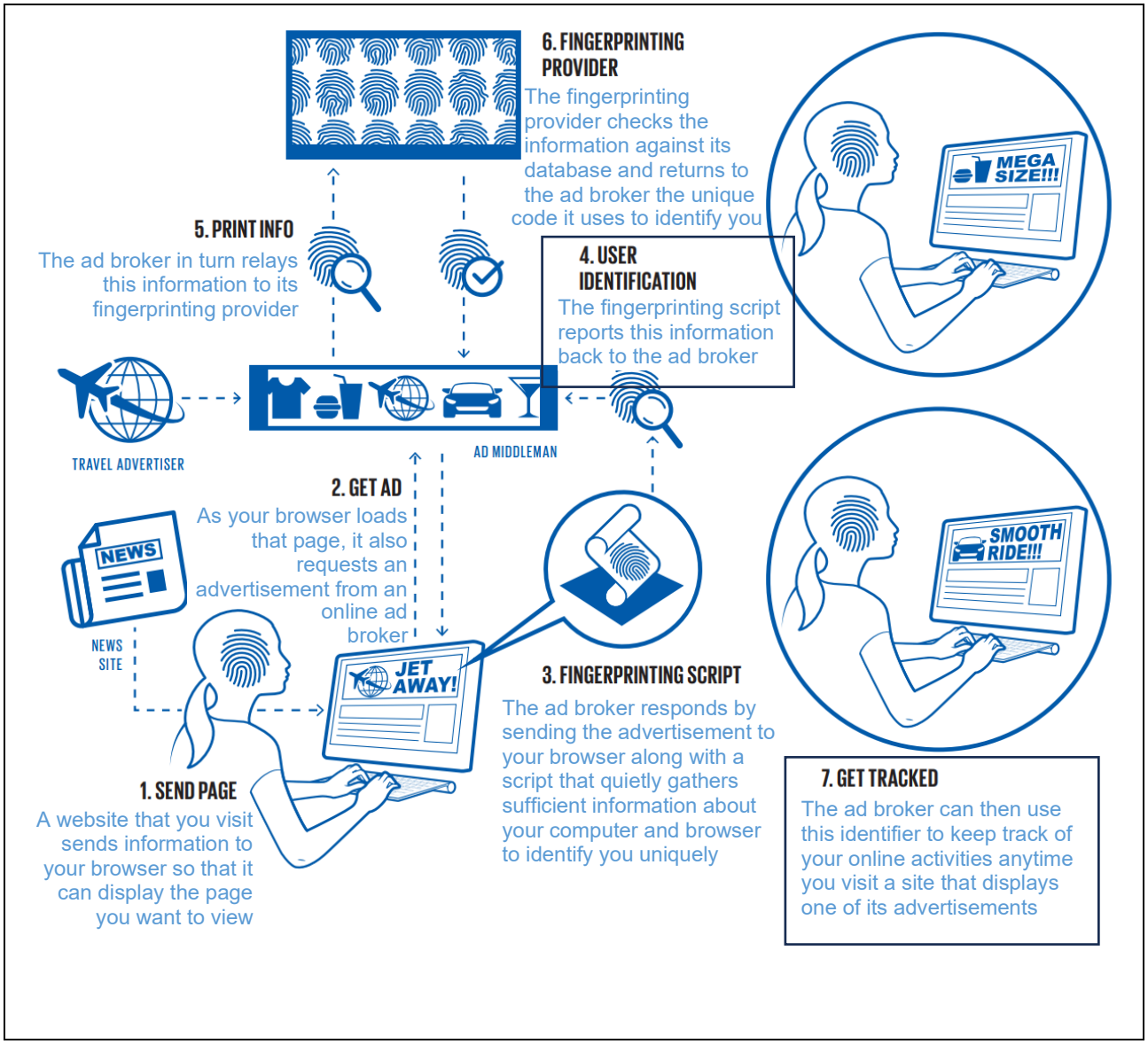

Figure 2: Diagram showing Role of Browser Fingerprinting in Online Advertising Network [23]

In contrast to stateful methods such as cookies, browser fingerprinting is stateless and does not require the storing of any information on the client-side browser. Moreover, browser fingerprinting possesses the ability to effectively circumvent the protections offered by private browsing mode of the browser [24]. As a result, it is widely regarded to be more privacy intrusive because browser fingerprints are opaque to the users and cannot be directly managed or removed by them [4]. Fortunately, it is reassuring that a user's browser characteristic information is also classified as personal data under GDPR. Hence, explicit user consent is required before they can be collected for the context of tracking and advertising. Nevertheless, it is common for website operators to proceed with the data collection, using the principle of "legitimate interest" as a justification for the exemption from consent requirements [4, 18].

As new web technologies are continuously being introduced, the same technology used for legitimate and essential web functionalities might also be used for fingerprinting [16]. For instance, the HTML5 canvas JavaScript API, originally intended to be used for dynamically rendering text and images on a browser, has also been repurposed to measure the subtle variations in how the GPU hardware renders graphic to generate a unique fingerprint of the device. While disabling JavaScript or restricting these APIs could effectively mitigate fingerprinting, it would also likely affect essential web functionality and result in a suboptimal browsing experience. Hence, striking a balance between browser fingerprint privacy protection and web usability remained a significant challenge.

Furthermore, due to the difficulties in identifying the intent and presence of browser fingerprinting on websites, as well as enforcing regulations around its use, researchers widely anticipate browser fingerprinting to remain as one of the prominent methods for user identification and tracking within the digital advertising industry [25].

2.2 Applications of Browser Fingerprinting

While Section 2.1 has provided a primer to the role of browser fingerprinting in digital advertising, its applications extend beyond this domain. This section discusses the other applications of browser fingerprinting, which can be broadly classified into two main categories: (a) Web Tracking and (b) Web Security [18].

2.2.1 Web Tracking

Web tracking is commonly referred to as the deliberate and systematic collection of online user data, regardless of user consent, for the purpose of constructing and monitoring user profiles over time [24, 26]. The use of web tracking in digital advertising to provide highly targeted and personalised advertisements has already been extensively discussed in Section 2.1.

2.2.1.1 Site Analytics and Performance

Website publishers might also use browser fingerprinting to monitor anonymous users on their websites for non-privacy invasive purposes, such as to support web analytics and performance testing to enhance site functionality and optimise user experience [27]. Web tracking enables website hosts to evaluate the performance of their sites by collecting and analysing visitor data. These

data could reveal valuable insights, not limited to the number of visitors over time, the proportion of new and returning visitors, the duration of site visits and the specific pages that attract the most attention. Furthermore, data pertaining to the users' device types and graphics rendering capabilities can assist website hosts in optimising their sites to enhance compatibility with the most prevalent devices and graphics processing configurations. By interpreting and leveraging these metrics, the hosts can refine the accessibility, content layout and design to attract visitors of similar profiles across diverse environments, thereby enhancing both visitation rates and overall user experience [28].

2.2.2 Web Security

Although browser fingerprinting is often linked to user privacy concerns, it can also serve as a constructive role in enhancing critical web security functions when applied in an ethical and transparent manner. Its security applications include the detection and mitigation of bot activities, fraud prevention, augmenting web authentication and identification of potentially vulnerable devices requiring patching.

2.2.2.1 Bot and Fraud Prevention

Bots are software-based applications designed and developed to automate and simulate human interactions with websites. While they facilitate large-scale web data collection by simulating human interactions in web measurement studies, they are also frequently exploited for malicious purposes such as credential abuse and advertisement fraud [29]. Through reverse engineering a commercial bot detection tool, Jonker [30] identified browser fingerprint attributes of a bot, including user agent, screen resolution and bot specific indicators like "webdriver", that can distinguish automated bots from a standard browser. By utilising this set of fingerprint elements, Jonker developed a bot-detection scanner to scan the Alexa Top 1 million websites for presence of bot detection scripts. The study revealed that 12.8% of these websites had exhibited evidence of employing browser fingerprinting for detecting bot activity.

Another application involves the validation of the integrity of the fingerprint itself. Due to the interdependent nature of the collected attributes, a change in one or more of the attributes might result in substantial deviations from the original fingerprint. By cross-referencing the browser fingerprints of incoming users against a database of verified authenticated devices, service providers can distinguish authorised users from potentially malicious actors. This additional fingerprint verification process not only reinforces the user authentication process but also enables the detection of fraudulent accounts on digital platforms. Several security firms such as ThreatMetrix, MaxMind and PerimeterX have already adopted browser fingerprinting for bots and anomalous activity detection [18]. This approach leverages the observation that bots often obscure their identity to avoid detection by frequently changing their attributes such as IP address, clearing cookies and randomising their browser device attributes. Consequently, their browser fingerprints are more unstable and less persistent than those of legitimate users.

2.2.2.2 Augmenting Web Authentication

Browser fingerprints also function as an additional layer of web authentication mechanism to secure online accounts [18]. Beyond traditional password-based verification, login systems can cross-reference the fingerprint of the login device against stored profiles to identify unfamiliar devices, thereby triggering the two-factor authentication to verify the login session [31]. For instance, Laperdrix et al. [32] developed a challenge and response protocol using a "dynamic fingerprinting scheme" based on canvas fingerprinting, which facilitates the differentiation between legitimate users and impersonators in a replay attack. Additionally, browser fingerprinting techniques can also further enhance CAPTCHA systems' resilience and robustness against bot-driven bypass attacks [33].

2.2.2.3 Identifying Vulnerable Devices

Browser fingerprinting technology can also assist system and network administrators in obtaining the current status of the hardware and software configuration across devices so that outdated or vulnerable components could be identified quickly, particularly within complex enterprise environments [34]. By collecting attributes such as operating system details, installed plugins, browser type and version, user-agent of the browsers, administrators can prioritise the deployment of software updates or security patches. This proactive approach would narrow the exposure window of the vulnerable devices to exploitation by attackers, thereby strengthening the overall security posture of enterprise networks [18].

2.3 Browser Fingerprinting Methods

Building upon the exploration of the applications of browser fingerprinting, this section presents a technical analysis of the underlying mechanisms that contribute to its effectiveness in user identification and tracking, with a particular focus on high-fidelity attributes that offers good distinguishing signals.

In the context of browser fingerprinting, constructing an accurate and reliable fingerprint for user identification requires a carefully curated selection and aggregation of device and browser attributes. An example of a set of attributes is shown in Figure 3 [35].The suitability of any given attribute is typically evaluated based on two fundamental criteria: entropy and stability.

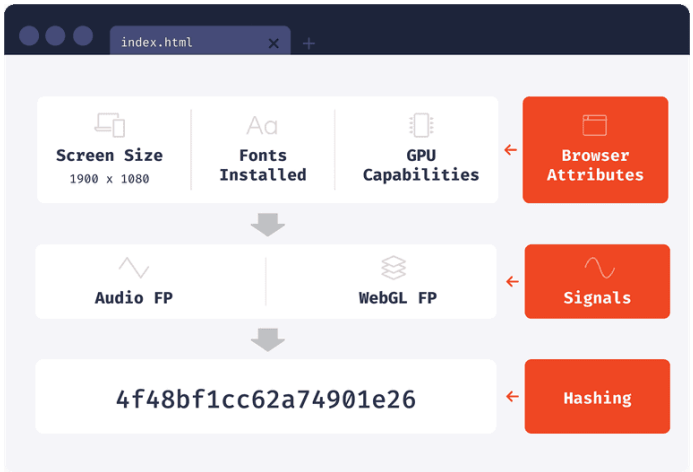

Figure 3: Diagram showing the set of attributes used to generate a browser fingerprint [35]

Entropy refers to the amount of distinguishing information an attribute exhibits across different user devices, contributing to the uniqueness of the final fingerprint. It is commonly measured using Shannon's entropy; however, a detailed discussion of this metric fall beyond the scope of the dissertation. Typically, attributes with higher entropy vary significantly between browsers, thereby increasing level of uniqueness and are more valuable for identification purposes [18]. For instance, supported browser fonts exhibit moderate entropy due to the diversity in the number, type and version of the fonts installed across different OS installations.

Stability, in contrast, refers to how consistent a device's attribute remained relatively unchanged over time. Highly stable attributes do not fluctuate frequently and is essential for long-term tracking and reliable user identification [20]. Screen resolution is an example of a highly stable attribute because a normal user would hardly change its screen resolution for optimum viewing experience.

Together, these two criteria would influence the overall accuracy and uniqueness of the final fingerprint for user identification over extended periods. Attributes that exhibit both high entropy and stability properties would reduce likelihood of false positives and false negatives, making them more suitable candidates for fingerprinting. Nevertheless, Zhao [9] demonstrated via his experimentation study that the aggregating of 476 traditional browser attributes could also yield highly accurate fingerprints, achieving a precision rate of 93.7%. His findings showed that high identification accuracy could also be attained if a sufficiently large set of traditional browser attributes with low to moderate entropy is used. Table 1 below illustrates a summary of the classification of some of the common browser attributes based on previous studies [20, 31, 36, 37].

Stability

Entrop	Low More False Positive	Moderate	High Less False Positive
Low More False Negative	battery	css	screen plugins language
Moderate		timezone	audio canvas font
High Less False Negative			webgl webrtc

Table 1: Classification of Browser Attributes Based on their Entropy and Stability [20, 31, 36, 37]

The subsequent sections examine the top five widely recognised fingerprinting attributes that are considered to be highly effective in generating distinctive and accurate user fingerprint.

2.3.1 Audio Fingerprinting

This fingerprinting method utilises the AudioContext JavaScript APIs to generate an audio signal with an oscillator node. The source signal can be further processed with different configuration nodes such as a dynamic compressor to add different variation and transformation, before sending the resultant signal to the destination for rendering. The raw signal can be extracted and hashed to create an audio fingerprint [38]. This fingerprinting process is illustrated in Figure 4.

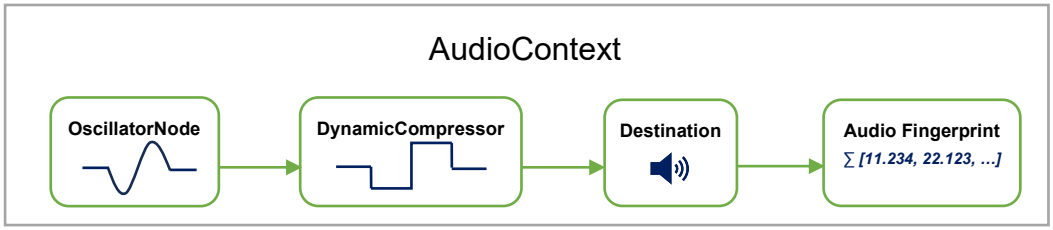

Figure 4: Diagram describing the process of creating an audio fingerprint using AudioContext

The same input audio signals processed on the same machine and browser would produce the same output, whereas processing done on differing hardware or software configurations may introduce subtle differences in the rendered audio content. This variations stem from the differences in the underlying hardware components, operating system and audio stack configuration and implementation. While Englehardt et al. [38] estimated the entropy of audio fingerprinting as moderate in their experiment study, the technique remained highly stable. This is due to the mathematical generation of both the input and output via a sequence of numbers. Slight inconsistencies in these sequence can be captured by the digital oscillator and analysed to create a more unique audio fingerprint representative of the device [39].

2.3.2 Canvas fingerprinting

The introduction of the HTML5 canvas element enabled websites to dynamically render graphics using JavaScript. It is primarily developed for displaying 2D graphics such as animations and real-time video processing. Notably, Mowery et al [36] first demonstrated how distinguishing information could be extracted from texts rendered in the canvas element, for fingerprinting devices through the toDataUrl method of the HTML5CanvasElement interface. The API returns a Base64-encoded representation of the raw binary pixel data. Their experimental analysis estimated that the entropy of canvas fingerprint is between moderate to high.

Similar to audio fingerprinting, the entropy associated with canvas fingerprint arise from the differences in each device's GPU hardware and driver implementations, causing browsers to render identical graphic with subtle distortions in the colour and shape. Furthermore, the canvas fingerprint is highly stable and remains consistent unless there is a modification to the underlying GPU hardware or software. All these rendering operations are executed silently in the background typically through a fingerprinting JavaScript embedded within a website, without the user's awareness. Owing to its accuracy due to a higher entropy and stability, canvas fingerprinting has become one of the most employed fingerprinting technique for user identification [39].

Figure 5: Example of a canvas fingerprint generated from canvas fingerprinting code of FingerprintJS

Figure 5 illustrates the different text and images rendered onto a canvas, alongside the raw data, which is converted into a data string, retrieved via the toDataUrl API. As shown, a diverse set of elements consisting of texts, emojis and geometrical shapes are drawn on the canvas to further increase the entropy of the raw data output.

2.3.3 Font Fingerprinting

Font fingerprinting utilises the observation that the set of installed fonts on individual devices tends to vary considerably across users. This variability arise from a combination of factors, including differences in default font libraries across various operating systems, custom fonts installed by users and their installation sequences [40]. However, following the deprecation of the Flash API, direct font enumeration and listing is no longer feasible. Current approaches predominantly rely on JavaScript-based font probing. This method involves the rendering of a predefined set of fonts on the browser and collecting associated font metrics such as text width and height. Variations in the dimensions or glyph of the font rendered, particularly the use of fallback fonts when the specified font is unavailable, serves as an indirect indicator of installed fonts. Due to the inherent limit of this partial enumeration technique, this approach could only provide a moderate level of entropy [37]. Nevertheless, it remains an accurate and stable fingerprinting technique as font configurations of users tend to remain stable over time [39].

Alternatively, Englehardt et al. [38] demonstrated that font probing could also be conducted through the HTML5 canvas element. Specifically, the CanvasRenderingContext2D interface provides a measureText method which returns text metrics, such as text width, that pertains to the rendered text. As illustrated in Figure 6 below, the full list of text width size metric associated with each probed font is collected. The variation in the rendered text width for each tested font contributes to the moderate entropy of the font fingerprint.

```
// Canvas font fingerprinting script.
Fonts = ["monospace" , ... , "sans-serif"];

CanvasElem = document.createElement("canvas");
CanvasElem.width = "100";
CanvasElem.height = "100";
context = CanvasElem.getContext('2d');
FPDict= {};
for (i = 0; i < Fonts.length; i++)
{
  CanvasElem.font = Fonts[i];
  FPDict[Fonts[i]] = context.measureText("example
      ").width;
}
```

Figure 6: Diagram demonstrating the pseudo code for canvas font fingerprinting [4]

2.3.4 WebGL fingerprinting

While the HTML5 canvas is primarily focused on rendering 2D graphics, the Web Graphics Library (WebGL) extends its capabilities by enabling the dynamic rendering of complex 3D graphics on the browser canvas. Similar to Canvas fingerprinting, WebGL fingerprinting involves the rendering of a predefined 3D graphic on a hidden canvas on the browser, with the resulting raw image data extracted via the toDataUrl method. The rendering process is inherently significantly influenced by the device's GPU hardware type and configuration, along with variations in the software implementation of the driver and rendering algorithms. These factors introduce significant entropy into the resultant 3D image, contributing to its uniqueness across devices [39]. The fingerprint is also highly stable and typically remain consistent over an extended period unless there are changes in the underlying GPU hardware or driver.

In addition to basic 3D rendering, Cao et al. [41] incorporated additional GPU-level features such as textures, varying, lights and models into the rendered image, thereby further contributing to the entropy of the final WebGL fingerprint. Their experimental findings also demonstrated that fingerprinting methods leveraging OS and hardware-level features, such as WebGL, also exhibit strong cross-browser stability on the same device. This suggests that WebGL fingerprints are largely independent of the browser type, and emerges to be a more privacy-invasive fingerprinting mechanism as compared to other conventional browser attributes-based methods.

2.3.5 WebRTC fingerprinting

Web-based Real-Time Communication (WebRTC) is a technology developed to facilitate direct peer-to-peer voice and video communications between browsers, eliminating the need for intermediary servers. To support communications between devices deployed behind private Network Address Translation (NAT) networks, WebRTC employs the Interactive Connectivity Establishment (ICE) protocol to facilitate NAT traversal through the dynamic discovery of optimal communication paths. During this process, WebRTC can reveal network-level details of the client such as public IP address, private IP address and local network. An example of how IP information could be retrieved from JavaScript is shown in Figure 7 [42].

```
var candidateRegEx = new RegExp(".*typ (host|srflx|relay).*");
var pc = new RTCPeerConnection({iceServers: [{urls: "stun:stun.l.google.com
    ↪ :19302"}]}, {optional: [{RtpDataChannels: true}]});
pc.createDataChannel("");
pc.createOffer(function(result){
 pc.setLocalDescription(result, function(){}, function(){});
}, function(){});
pc.onicecandidate = function(iceCandidate){
 if(iceCandidate.candidate){
  var c = iceCandidate.candidate.candidate;
  console.log("Detected IP " + c.split(" ")[4] + " as " + candidateRegEx.exec(c)
      ↪ [1]);
 }};
```

Figure 7: Code snippet on how to get IP information in the browser [42]

Critically, these information can be silently accessed in the background via JavaScript, making them useful for fingerprinting and identifying users, even when privacy-preserving tools such as proxies or VPNs are used [20, 38]. A study conducted by Reiter et al. [42] demonstrated the high uniqueness and stability of the WebRTC fingerprint data, revealing that 97% of the 80 devices tested could be reliably distinguished.

2.4 Browser Fingerprinting Countermeasures

The capabilities of modern web browsers have evolved substantially with the advancements in web technologies such as the introduction of HTML5. However, these capabilities have inadvertently facilitated more invasive tracking techniques via browser fingerprinting, which pose risks to user privacy. In response, various countermeasures have been proposed and implemented either natively within the browser or via external anti-tracking browser extensions to mitigate the threats of browser fingerprinting, with the latter as the more prominent approach.

The protection strategies can be broadly categorised into three main approaches: (a) Randomisation, (b) Normalisation, and (c) Heuristic [4], each having their own advantages and trade-offs. Ultimately, it is evident that there is no single methodology that can achieve an optimal balance between safeguarding privacy protection and maintaining usability.

2.4.1 Randomisation

Randomisation strategies aim to introduce variability into browser attributes, such as injecting noises into return values of APIs, to reduce consistency across different browser sessions. This technique reduces the attribute's stability by hiding and hindering the ability of client-side APIs to capture the real user environment attribute [43]. Laperdrix et al. [44] developed the browser prototype FPRandom from Firefox code base to demonstrate that minimal noise can be injected into canvas and audio fingerprinting attributes to mitigate tracking with negligible impact to the overall user experience.

Conversely, Vastel et al. [45] demonstrated that the ability of randomisation-based countermeasures in restricting fingerprinting is not only limited, but also detectable and paradoxically indicating the presence of countermeasures, thereby distinguishing protected users from normal browsing behaviours [4, 45]. Moreover, another limitation lies in their susceptibility to reverse-engineering when the applied noise algorithm is predictable. A notable example is Safari 17's Advanced Tracking and Fingerprinting Protection (ATFP) mechanism,

which injects minimal noise into the Web Audio API output. The noise algorithm, however, was proven to be mathematically reversible and denoised to recover a stable fingerprint. [46].

2.4.2 Normalisation

The normalisation approach seeks to ensure that all browser instances display similar characteristics, thereby reducing the attributes' entropy. This is typically done by either disabling access or spoofing the return values of specific APIs [4]. The Tor browser has adopted normalisation as its mitigation strategy, making 24 modifications to its browser. These include disabling the canvas and WebGL APIs, removing plugin support and standardising the available set of fonts [18]. While this mitigates fingerprinting to a certain extent, it limits the functionality of the website, resulting in the degrading of user experience [47]. A study conducted by Khattak et al. [48] presented clear evidence of blocking Tor on the web, amounting to 3.67% of the top 1000 Alexa sites. Similar to the randomisation approach, normalisation can also inadvertently reveal the use of Tor browser itself. In environments where user base is limited or when individuals deviate from the default configuration, Tor browser's distinct characteristics may become a unique identifier for tracking [11].

2.4.3 Heuristic

Heuristics are rule-based methods, typically relying on predefined patterns in filtering rules to identify fingerprinting techniques in scripts. These heuristics are meticulously crafted to maintain a low false positive rate. However, this precision comes with the downside of an overly stringent detection criteria, whereby slightly modified versions of known fingerprinting scripts can evade detections [4]. Moreover, rule-based detection techniques often suffer from limited detection coverage and their effectiveness is heavily dependent on the expertise of the rule author [47]. Furthermore, such heuristic-based detection methods could be easily bypassed, thereby demanding continuous maintenance to remain effective against emerging and evolving fingerprinting techniques.

The most prevalent form of heuristics-based protection mechanism involves the blocking of network requests from websites that matched the entries in curated URL filter lists, such as EasyList and EasyPrivacy [18]. This blocking mechanism is widely regarded as both performance efficient and highly effective. Loading of browser fingerprinting scripts from known domains can be blocked with minimal operational overhead and lower false negatives. Such functionalities are commonly implemented through privacy-focused browser extensions like Ghostery and uBlock Origin, leveraging native browser network APIs such as declarativeNetRequest API in MV3 to intercept and block network requests [14]. In addition to third-party browser extensions, mainstream browsers such as Edge and FireFox have also integrated heuristic-based mechanisms; incorporating the Tracking Protection List (TPL) developed by Disconnect as part of their anti-tracking protection capabilities [47].

However, a notable limitation lies in its longer lead time needed to update these filter lists with newly identified fingerprinting domains [49, 50]. To enhance detection adaptability, filtering rules often incorporate regular expression syntax

to accommodate variations in URL and script identifiers that might otherwise bypass static filtering methods.

3 Related Work

3.1 Origins of browser fingerprinting

Mayer first demonstrated the proof-of-concept of browser fingerprinting in 2009 with his experimental study on the deanonymisation of web clients due to underlying differences in browsing environments such as the OS, hardware and browser configurations [51]. Over a 2-week period, a set of properties of the built-in JavaScript objects of browsers, such as navigator, screen, plugins, and MIME types, were collected to generate fingerprints from the browsers of each visitor to a blog page. Among the 1328 visitors, he found that 96.23% could be uniquely identified. However, due to the experiment's limited scale and duration, it was unable to establish a high degree of statistical confidence that browser environment properties could be used to deanonymize web clients. Another limitation of the experiment is its inability to measure the extent to which each individual property contributed to the fingerprint's uniqueness, as the hashing process results in the loss of detailed information.

In the following year, Eckersley et al. [40] conducted a larger-scale study, Panopticlick, to investigate the degree to which modern browsers are subjected to fingerprinting. He implemented a browser fingerprinting algorithm by concatenating measurement strings from a collection of 8 browser attributes: User Agent, HTTP ACCEPT headers, cookies enabled, screen resolution, time szone, browser plugins, system fonts and partial supercookie test. Among the 470,161 fingerprints collected over a 2-week period, 83.6% of fingerprints were unique. Even though the experimental results indicated that 37.4% of revisiting users over a period of more than 24 hours had at least one fingerprint change, it was possible to correlate the different fingerprints of the same user using simple heuristics with 99.1% correct guesses and 0.86% false positive rate. These findings ultimately highlighted the feasibility of browser fingerprinting in real-world scenarios.

3.2 Prevalence and Scope of browser fingerprinting

Nikiforakis et al. [10] conducted the first study in 2013 on the real-world commercial adoption and the workings of fingerprinting on the Internet by analysing the usage of fingerprinting libraries of three large companies: BlueCava, Iovation and ThreatMetrix. A taxonomy of all possible scope of attributes and techniques from the fingerprinting libraries of each of the three companies was created. The taxonomy highlighted the inherent flexibility and diversity in fingerprinting attributes and methods. The absence of a universally accepted framework or regulatory standard for browser fingerprinting greatly complicates the implementation of effective and sustainable user privacy safeguards against them. Nikiforakis et al. then crawled the Alexa top 10,000 sites and discovered that 40 sites (0.4%) were utilising the fingerprinting code from the three commercial providers. While the study confirmed the use of browser fingerprinting on some of the most widely visited websites, it was restricted to the detection of fingerprinting techniques used by the three companies and thus, not a good representation of the broader adoption level across the Internet.

In the same year, Acar et al. [16] developed FPDetective, a framework designed to detect and analyse browser fingerprinting using JavaScript and Flash-based font probing. Flash-based font probing was chosen as the key detection factor because of its high entropy at 17.1 bits, making it a necessary feature for browser fingerprinting [16, 40]. In contrast to Niforakis et al. ‘s library-specific approach, FPDetective uses a feature-dependent approach to identify fingerprinting. A website would be classified as performing fingerprinting if it loaded more than 30 system fonts, probed the navigator and screen properties and enumerate plugins or MIME types. After scanning the Alexa top 1 million websites with FPDetective, 13 instances of JavaScript fingerprinting libraries were found on 404 (0.04%) websites, whereas 145 (1.45%) sites of the Alexa top 10,000 sites were found to leverage Flash-based fingerprinting. The research findings reaffirmed the significantly higher adoption level of fingerprinting than earlier studies. Notably, while the companies responsible for deploying fingerprinting could be identified, it is not possible to effectively determine the purpose of their actual usage.

In 2014, Acar et al. [52] conducted the first study of real-world adoption of canvas fingerprinting practices. Using a feature-dependent methodology similar to FPDetective, the execution of the toDataURL API was used to identify presence of canvas fingerprinting. Further manual analysis was carried out to identify 3 filtering criteria for detection: (a) both toDataURL and fillText calls must come from the same URL, (b) the canvas image size must be at least 16x16 pixel and contain multiple colours, and (c) the image data must be requested in lossless format. After scanning the Alexa top 100,000 sites, 20 domains were discovered to be actively using canvas fingerprinting techniques on 5,542 (5.5%) of them. An interesting observation by Acar et al. was that regardless of user consent, canvas fingerprints were still being collected by AddThis and Ligatus providers. This finding raised concern about the effectiveness of user consent mechanisms in addressing privacy concerns.

In 2016, Englehardt et al. [38] developed the web privacy measurement tool, OpenWPM, to investigate the prevalence of device fingerprinting at scale. In addition to the detection of canvas fingerprinting, two new detections methods were introduced to identify canvas font and WebRTC fingerprinting. Scanning the Alexa top 1 million sites revealed that 14,371 (1.6%) of them served canvas fingerprinting scripts from 400 different domains. In contrast, canvas font and WebRTC-based fingerprinting were found to be notably lower, appearing at 3,250 (0.325%) and 715 (0.0715%) websites respectively. The study also established the methodology for the semi-automated discovery of novel fingerprinting techniques by tracking and observing new API usage patterns from known fingerprinting scripts. With this approach, new fingerprinting techniques leveraging AudioContext and Battery Status APIs were discovered.

As previous works on browser fingerprinting detection were primarily heuristics-based and carefully crafted to minimise false positives, it is essential to maintain and update the detection mechanism continuously to keep pace with the rapid evolution of fingerprinting techniques. Iqbal et al. [4] proposed the development of FP-Inspector, a “ML-based syntactic-semantic approach” to detect browsing fingerprinting using a combination of static and dynamic analysis. Using FP-

Inspector to scan across the Alexa top 100,000 websites, 2,349 domains were found hosting fingerprinting scripts for 9,040 (10.18%) websites. Compared to 2016, the prevalence of adoption had increased significantly. Iqbal et al. [4] further observed that fingerprinting was not solely employed for tracking purposes, but also served as a mechanism to detect ad fraud, with three of the top five vendors specialising in verifying the authenticity of ad impressions. Additionally, the study highlighted that 17.2% of the fingerprinting vendors engaged in cookie syncing, often in collaboration with well-known AdTech companies. These findings underscore the complexity in identifying the true intent behind the applications of browser fingerprinting, whether it was employed for user tracking or web security.

Recent studies [9, 53, 54] have redefined the traditional monitoring and analysis of static and dynamic execution of attribute-based API executions for detecting browser fingerprinting into flow-based runtime taint-propagation models. These models aim to reduce the false positives associated with the legitimate use of Canvas and WebGL APIs for their intended purposes. A valid fingerprinting instance is characterised by a propagation flow originating from a taint source, defined as the collection of browser attributes, and terminating at a taint sink, which may correspond to the remote transmission of these data. Li et al. [54] used the bytecode instrumentation approach to develop FPFlow, a dynamic taint analysis framework that monitored the data flow from the retrieval of any of 17 browser attributes via Document Object Model (DOM) APIs to their remote transmission, achieving a low false positive rate of 9.72%.

In contrast, Zhao [9] expanded the concept of a taint sink to encompass any objects which tainted sources aggregated. His framework, FProbe, achieved an even notable low false positive rate of 0.92%. A comparative analysis of findings between Zhao and Iqbal et al. [4] showed a decline in the percentage of detected fingerprinting activity among the Alexa top 100,000 visited websites from 10.18% to 7.33%. This discrepancy is likely attributable to the discontinuation of the Alexa ranking service since mid-2022, which not only impacted the composition and representativeness of the dataset used for evaluation, but also its ability to accurately reflect the current prevalence of browser fingerprinting in the digital landscape.

3.3 Browser fingerprinting Countermeasures

Acar et al. [16] analysed the effectiveness of fingerprinting protection of 2 tools: Tor Browser and FireGloves, a proof-of-concept Firefox extension for research purposes. Overall, weaknesses caused by imperfect implementation of mitigation measures were found to be exploited to bypass protections. Additionally, these might inadvertently make the devices more identifiable. The lack of support and maintenance for the proof-of-concept extension is also an area of concern, highlighting the challenge and complexity of implementing robust and effective protection measures.

Englehardt et al. [38] evaluated the effectiveness of two tracking protection tools: Disconnect and the combination of EasyList and EasyPrivacy. Both tools' main protection mechanisms are based on their curated blocking lists. The test results indicated that, while both tools performed similarly and were effective in

blocking prominent fingerprinting scripts, they were ineffective against less popular ones. The limitation of their blocking abilities was especially apparent in scripts utilising lesser-known methods, such as WebRTC and Audio fingerprinting. This was because the curation of the blocking lists is heavily dependent on manual analysis. They were also ineffective in restricting fingerprinting scripts that are served from first party domains and CDNs [4].

Ukani [11] further classified two major browser fingerprinting countermeasures approaches into four subcategories: Randomisation - comprising (a) entropy reduction, (b) entropy enhancement, and Heuristics - (c) blocking and (d) alert visibility enhancement. While each of these four sub-approaches has its own advantages and disadvantages, the approach of increasing alert visibility was said to be the least effective as it did not prevent fingerprinting directly and caused more confusion for users. Conversely, the blocking approach was shown to be the most effective since it blocked fingerprint scripts from known trackers with low false positives.

Ultimately, there is a general consensus that there is no single universal method that can effectively prevent browser fingerprinting without adversely impacting user experience. The best way forward likely involves a hybrid strategy that combines multiple countermeasure approaches.

3.4 Summary

In this study, we conduct a large-scale measurement study of browser fingerprinting across the Tranco Top 10,000 websites to identify its prevalence, particularly in the context of user identification for digital advertising. We then introduce FPAnalyze, a dynamic analysis-based heuristic framework designed to detect two categories of browser fingerprinting, invasive and general, as well as the spectrum of fingerprinting techniques that are deployed in typical browser fingerprinting JavaScript code. Finally, we evaluate the effectiveness of five widely used browser extensions in mitigating the threats posed by invasive browser fingerprinting for user identification.

4 Design Methodology

This section presents the ethical considerations; design rationale and methodological framework used in the study. A descriptive quantitative research method is adopted to systematically collect and analyse surface web data, and provide an accurate evidence-based assessment on the prevalence of browser fingerprinting. A high-level overview summarising the design workflow is outlined in Figure 8.

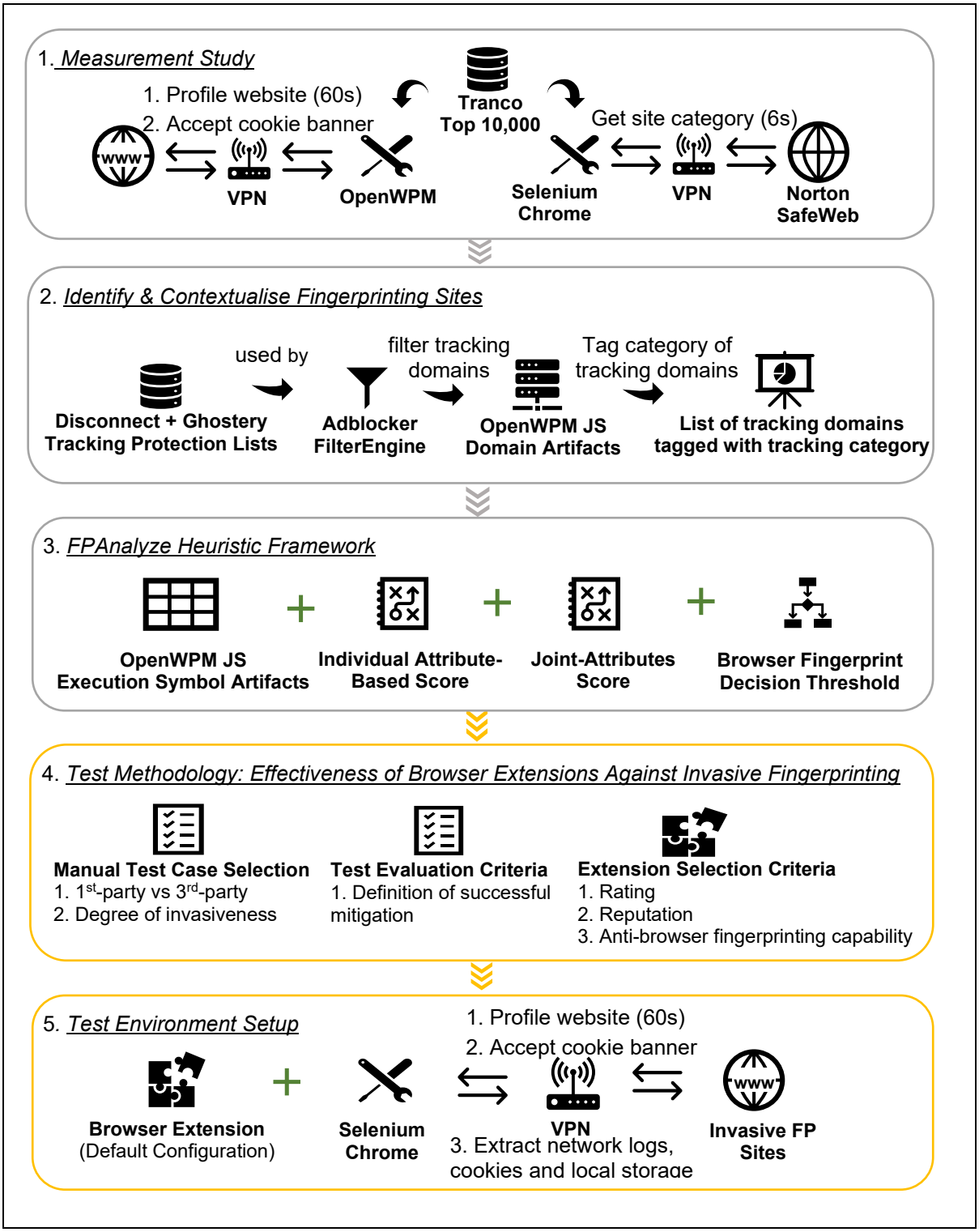

Figure 8: Overview of Experimental Design Architecture

First, we would conduct a measurement study of the Tranco top 10,000 most visited websites on the surface web to identify the prevalence of browser fingerprinting, particularly in the context of user identification in digital advertising.

Next, to contextualise these fingerprinting practices, we use tracking protection lists from two commercial vendors: Disconnect and Ghostery, to classify script domains based on their intended purpose and their engagement in invasive or general fingerprinting activities.

Furthermore, we propose FPAnalyze, a dynamic analysis-based heuristic framework designed to detect the two categories of browser fingerprinting: invasive and general as defined in Section 1.4. This framework combines the feature-dependent methodology of Englehardt et.al [38] with a multi-threshold browser attribute scoring system that evaluates the likelihood of co-occurrence of specific groups of fingerprinting techniques. We deploy FPAnalyze to the measurement study dataset to identify typical combinations of browser attributes collected by the two categories of fingerprinting scripts.

Finally, we setup the test environment to evaluate the effectiveness of five popular browser extensions in mitigating invasive fingerprinting. This assessment spans across 40 representative test cases derived from our measurement study dataset. Specifically, two areas are being examined: (a) whether browser extensions exhibit comparable effectiveness in countering fingerprinting attempts originating from first-party web publishers versus third-party service vendors, and (b) whether the degree of invasiveness of fingerprinting techniques, measured by FPAnalyze's heuristic score, influences the mitigation capabilities of these extensions.

4.1 Ethical Consideration

Given the unpredictable nature of websites accessed during web profiling via the campus Wi-Fi network infrastructure, all web profiling activities conducted are routed through a Virtual Private Network (VPN). This ensure data transmission is secure and mitigates reputational risks associated with an institutional traffic interacting with malicious or questionable websites. To further mitigate the risk of exposure to malware infection from compromised or malicious websites, web profiling is conducted within a Virtual Machine (VM) environment. This setup enables the creation of system snapshots, facilitating the restoration of prior states following each profiling session as a precautionary measure.

The data collection process employed in this study is largely passive. Only the landing page of each website is downloaded for analysis, with simulated user interactions limited to basic actions such as mouse movements, scrolling and clicking on cookie consent banner. For repeated queries such as submission of hostnames to the Norton Safe Web service for site categorisation, the study strictly adheres to the publisher's terms and conditions outlined in the robots.txt file, if available. Additionally, request throttling mechanisms are implemented to minimise server load and prevent unintentional denial-of-service occurrences.

4.2 Web Measurement Study using Instrumentation Tool OpenWPM

This study would profile the top 10,000 successfully visited websites from the Tranco list [55] to balance representativeness of website traffic and the practical constraints of time and computational resources. Although previous studies commonly relied on the Alexa Top 1 million list maintained by Amazon, its official discontinuation in 2023 [56] renders it unsuitable as a current benchmark of web usage trends. In contrast, the Tranco list provides a research-oriented alternative that is hardened against manipulation and widely cited in academic literature, making it a reliable and representative dataset [57, 58]. Its versioned and timestamped dataset ensured researchers can replicate experiment with the exact same set of datasets.

4.2.1 Extending Capabilities and Configuration Settings of OpenWPM

OpenWPM [38] is chosen as the web instrumentation tool because its technology is mature and actively maintained. Its reliability is well-established, having been widely adopted in numerous web privacy and measurement research studies [57]. Furthermore, its integration with the Tranco list facilitates automated and seamless selection of representative websites, thereby streamlining the initial profiling process.

However, some websites have been observed to suppress web content or trigger CAPTCHA to mitigate access by automated clients such as OpenWPM. Krumnnow et al.'s [58] reported that approximately 14.7% of the landing pages in Tranco's Top 100K list incorporated bot detection scripts capable of identifying OpenWPM. To circumvent these mechanisms, this study configures and extends OpenWPM's features to reduce its detectability. Specifically, web profiling is conducted using native GUI mode, which exposes the lowest detection surface compared to its alternative headless modes. Moreover, dynamic browser screen configuration is implemented, with a default resolution set to 1920 x 1280 (Full High-Definition) to emulate a typical standard user environment. To better emulate human behavioural patterns, the bot mitigation routine was enhanced to incorporate randomised bidirectional webpage scrolling, interleaved within mouse movements.

To optimise profiling efficiency, each website is allocated a maximum loading time of 60 seconds, beyond which a timeout is enforced. Additionally, a new automated command is implemented to identify and accept consent banners on privacy-compliant tracking sites, thereby improving the accuracy of browser fingerprinting prevalence estimates for user identification [14, 59]. During the profiling process, all retrieved first-party and third-party JavaScripts and HTML page sources are stored in OpenWPM's LevelDB database to facilitate subsequent analysis and validation. The configuration settings applied to OpenWPM are presented in Table 2.

browser parameters	type	configuration
http_instrument	built-in	True
cookie_instrument	built-in	True
navigation_instrument	built-in	True
js_instrument	built-in	True
display_mode	built-in	native

save_content	built-in	script, main_frame, subframe
screen resolution	custom	(1920, 1080)
cookie consent	custom	“accept”

Table 2: OpenWPM configuration used for Profiling

As OpenWPM instruments only a limited subset of JavaScript APIs by default, its instrumentation scope was further customised and expanded to capture additional fingerprinting attributes. The customisation was largely based on API configurations defined in FP-Inspector [4]. Furthermore, the call stacks associated with instrumented APIs are also logged to facilitate subsequent static analysis and validation of fingerprinting scripts. A comprehensive list of these configurations is provided in Appendix A.1.

4.2.2 Filtering Successful Visits and Categorisation of Websites

To enhance representativeness despite the constrained sample size, the dataset is purposefully cleaned to isolate and analyse the top 10,000 successfully visited domains from the Tranco list. A visit is deemed successful if no exceptions occurred during profiling, with the website fully or partially loaded before hitting the 60-second timeout threshold.

The validated websites are then enriched with site category metadata using the classification scheme provided by Norton Safe Web [60]. This service, provided by a globally trusted cybersecurity company, has also been used in prior study for website categorisation [58]. Given that individual sites can be assigned multiple categories, all assigned categories are tallied to facilitate comprehensive categorical analysis. This enrichment helps explore how different types of websites may be more likely to favour browser fingerprinting to identify users.

4.3 Identifying and Contextualising Fingerprinting Websites

For each profiled website and its associated script domain instances, the open-source Ghostery adblocker [2] javascript FilterEngine library is used in combination with the Disconnect and Ghostery Tracking Protection Lists for matching. To mitigate the limited coverage inherent in relying on a single source, these two reputable tracking protection lists are integrated to improve coverage of known tracker domains. The FilterEngine library is chosen because it functions like an in-browser ad blocking mechanisms when provided with relevant information such as source website and script domains, thereby ensuring both consistency and accuracy in detection [61]. Following a successful domain match, the script domain is mapped to its corresponding tracker category, as specified in the respective Tracking Protection Lists. In instances where the lists provided conflicting classifications, a resolution protocol is developed to determine the final category assignment.

4.3.1 Disconnect and Ghostery Tracking Protection List (TPL)

The study leverages the domain expertise of Disconnect and Ghostery for three key reasons. First, both entities are among the few privacy-focused companies that maintain publicly accessible Tracking Protection Lists, which categorise

[2] https://github.com/ghostery/adblocker/tree/master/packages/adblocker

trackers based on their intended functions, such as advertising, site analytics or anti-fraud. Second, they provide classification transparency by detailing the methodologies used to determine each tracker's specific capabilities and purposes [17, 62]. Third, their dataset have been widely adopted in prior web measurement studies such as FP-Inspector [4], FP-tracer [53] and investigations into third-party web tracking [63]. By incorporating these resources, the study enhances its ability to accurately contextualise and classify fingerprinting behaviours observed in real-world surface web [38]. The tracker classification schema for Disconnect and Ghostery are presented in Table 3 and Figure 9 respectively.

Categories	Description
Advertising	A tracker which also displays or enables ads or marketing offers. These types of ads can track your personal information and expose you to malware, even if you don't interact with them.
Analytics	A tracker which collects your information and may build a profile based on your online activity that can be connected with your real name or other unique identifier.
Anti-Fraud	A tracker may be classified as anti-fraud if its explicit purpose is to prevent or detect fraud and does not utilize data collected in a third-party context (including IP addresses and user identifiers) for any purpose not directly related to fraud detection or prevention, including the use and sharing of such data to enable tracking of particular users or devices by other services.
Consent Managers	A tracker may be classified as a Consent Manager if its explicit and primary purpose is to manage consent preferences and does not utilize data collected in a third-party context (including IP addresses and user identifiers) for any purpose not directly related to managing consent preferences, including the use and sharing of such data to enable tracking of particular users or devices by other services.
Fingerprinting	A tracker may be classified as a fingerprinter if it identifies particular users or devices based on the properties of the browser, device, network, or any other properties of the computing environment, without using client-side storage of cookies or other data. We differentiate between two sub-categories of fingerprinters: • A tracker may be classified as a general fingerprinter if it uses browser or device features or properties in unintended ways to identify and track a particular user or device. • A tracker may be classified as an invasive fingerprinter if it uses an API to extract information about a particular user's computing environment when the API was not designed to expose such information.

Table 3: Table of Disconnect Tracker Categories and Description[3]

[3] https://disconnect.me/trackerprotection#categories_of_trackers

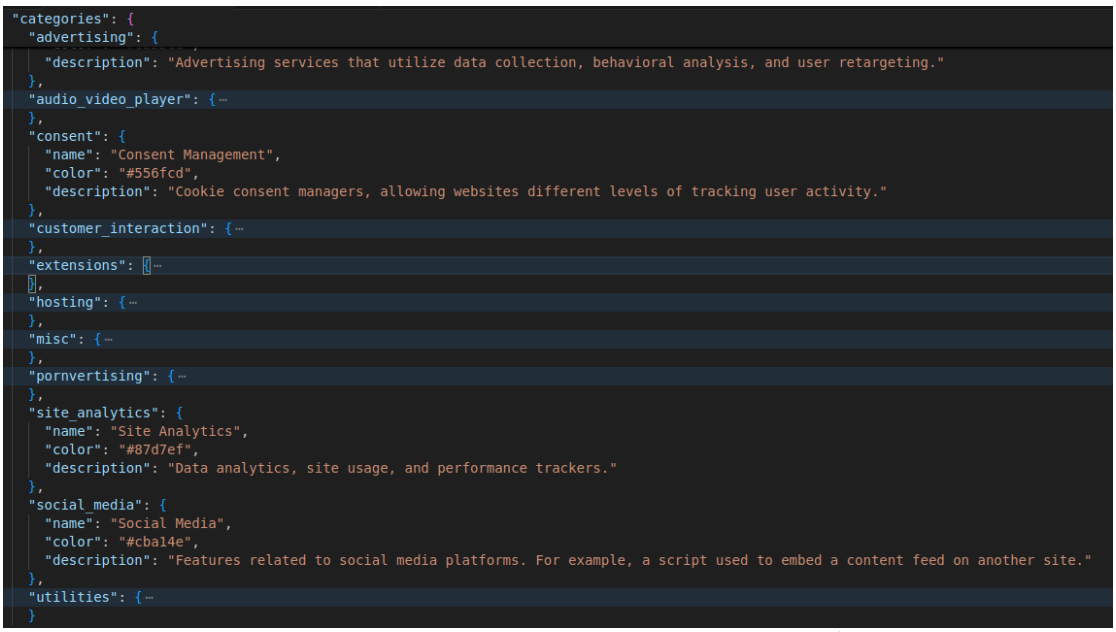

Figure 9: Classification of trackers used by Ghostery[4]

4.3.2 Normalising and Assigning Category Tags

Since each company maintains its own categorisation schema and naming conventions, it is essential to develop a normalisation protocol to standardize the category tags. This process began with the aggregation of all unique category labels from each list, followed by a comparative analysis to identify semantic category overlaps. Semantically similar categories are grouped together under unified headings. The protocol also accounts for specific peculiarities in the Disconnect dataset, wherein a tracker domain may be assigned dual classifications, such as "FingerprintingInvasive" or "FingerprintingGeneral" alongside broader categories such as "Advertising". To preserve the fingerprinting context, a prefix of "FpI_" or "FpG_" is prepended to Disconnect's tracker categories when a domain is simultaneously classified under "FingerprintingInvasive" or "FingerprintingGeneral" alongside another category. The complete set of normalised tags is presented in Table 4.

Disconnect	Ghostery	Normalised
Email	-	email
Email Aggressive	-	email_agressive
Advertising	advertising	advertising
Analytics	site_analytics	analytics
FingerprintingInvasive	-	fingerprintinginvasive
FingerprintGeneral	-	fingerprintgeneral
Anti-fraud	-	anti-fraud
Social	social_media	social
ConsentManagers	consent	consent
Cryptomining	-	cryptomining
-	audio_video_player	audio_video_player
-	customer_interaction	customer_interaction
-	extensions	extensions
-	hosting	hosting
-	misc	misc

[4] https://github.com/ghostery/trackerdb/tree/main/db/categories

-	pornvertising	pornvertising
FpG_*	-	fpg_*
FpI_*	-	fpi_*
-	-	uncategorised

Table 4: Overview of Category Tags Classification

4.3.3 Resolving Category Conflict

In the event of category conflicts between the two datasets, the following resolution protocol is applied in sequential order:

1. *Primary Vendor Precedence*: Disconnect's label takes precedence by default, due to its more elaborated focus on browser fingerprinting and transparent documentation of evidence-based classification rationale [17]. However, if the label belongs to one of the following lower-context classifications: "FingerprintingInvasive", "FingerprintingGeneral", "Email" or "Email Aggressive", a comparative evaluation with Ghostery is activated.

2. *Contextual Comparison Logic*: If Ghostery's classification falls under one of the following contextual labels: "misc", "utilities" and "uncategorised", Disconnect's label is retained; otherwise Ghostery's label overrides.

3. *Prefix Inheritance*: In instances where Ghostery's label is selected and the original Disconnect label falls under the Fingerprinting category, the prefix "FpI_" or "FpG_" is inherited to preserve the fingerprinting context for subsequent analysis.

4.4 FPAnalyze Heuristic Framework

This section presents the design and implementation of FPAnalyze, a dynamic analysis-based heuristic framework inspired by the key insight from Iqbal et al. [4]: "browser fingerprinting scripts typically do not use a technique (e.g., canvas fingerprinting) in isolation but rather combine several techniques together." FPAnalyze employs a dual-stage scoring and threshold mechanism, based on assessing both individual and joint JavaScript API and attributes to determine the presence of specific fingerprinting techniques and decide if their collective usage constitutes fingerprinting behaviour. The framework also incorporates a novel approach by leveraging bot detection techniques as indicators of fingerprinting behaviour. This integration recognises that certain bot detection signals can serve as proxies for identifying scripts engaged in browser fingerprinting activities.

4.4.1 Technical Definitions of Invasive and General Fingerprinting

With reference to Section 1.4, this section first establishes the formal technical definitions and classification criteria for both invasive and general fingerprinting techniques within the framework. These definitions serve as the basis on the scoring and threshold mechanisms used to evaluate fingerprinting behaviour.

Invasive Fingerprinting - A fingerprinting script is classified as invasive if it employs (a) more than one attribute from the critical group, or (b) at least one attribute from the critical group, excluding bot detection, in combination with two or more attributes from the general group as defined in Table 5.

General fingerprinting – A fingerprinting script is classified as generic if it employs bot detection mechanism alongside at least two additional attributes from the general group as defined in Table 5.

This conservative classification strategy is designed to minimize false positive rate, thereby ensuring a more reliable ground truth for identifying fingerprinting behaviour.

4.4.2 Individual Attribute-Based Heuristic Scoring and Threshold

The individual attribute-based heuristic scoring mechanism quantifies the likelihood that a script uses a specific browser fingerprinting technique. This enables a nuanced understanding of the spectrum of fingerprinting methods, whether invasive or general, that might be deployed concurrently within a single script. To effectively manage the diverse range of attributes and APIs involved, fingerprinting techniques are organised into two primary groups: (a) the critical group, which indicates invasive fingerprinting such as canvas and WebGL, and (b) generic group, comprising group attributes such as browser metadata and storage access, which are more likely to serve legitimate purposes and are less-indicative of fingerprinting intent. These two groups are further divided into 11 sub-categories detailed in Table 5.

A notable exception is the inclusion of bot detection behaviours within the critical group. While bot detection is widely studied in its own domain [58, 64, 65], it has not been prominently featured in prior browser fingerprinting research. This framework incorporates bot detection attributes in recognition of their relevance to web security and the hypothesis that advertisers also seek to differentiate genuine user interactions from automated bot activity.

Group	Subcategories
Critical	audio, canvas, font, webgl, webrtc, detectbot
Generic	browser, system, storage, screen, transmission

Table 5: Classification of Attributes in Attribute-Based Heuristics Scoring

For each subcategory, the framework selects the top eight specific attributes or APIs most commonly accessed, arranged in descending order of importance based on prior literature [4, 9, 38, 65]. These attributes are encoded using a bitmask scheme which serves two key purposes: (a) enabling weighted scoring by assigning positional significance to each attribute, and (b) providing explicit indicators of which attributes are detected during script execution. This weighted scoring is important in the critical group because invasive techniques often use multiple combinations of APIs to add entropy to the fingerprinting technique. The symbol artifacts of each loaded scripts are matched against the predefined subcategory configuration outlined in Figure 10.

```
FP_ATTRIBUTES = {
    'audio':    ['OfflineAudioContext.oncomplete', 'OfflineAudioContext.createOscillator',
                 'OfflineAudioContext.startRendering', 'OfflineAudioContext.createDynamicsCompressor',
                 'OfflineAudioContext.createAnalyser', 'OfflineAudioContext.createBiquadFilter'],
    'canvas':   ['HTMLCanvasElement.toDataURL', 'CanvasRenderingContext2D.fillText',
                 'CanvasRenderingContext2D.fillStyle', 'CanvasRenderingContext2D.isPointInPath',
                 'CanvasRenderingContext2D.strokeText', 'CanvasRenderingContext2D.strokeStyle'],
    'font' :    ['CanvasRenderingContext2D.measureText', 'CanvasRenderingContext2D.font'],
    'webgl':    ['window.WebGLTexture.toString', 'window.WebGLShader.toString',
                 'window.WebGLShaderPrecisionFormat.toString', 'window.WebGLRenderingContext.toString'],
    'webrtc':   ['RTCPeerConnection.onicecandidate','RTCPeerConnection.createDataChannel',
                 'RTCPeerConnection.createOffer', 'RTCPeerConnection.setLocalDescription'],
    'screen':   ['HTMLCanvasElement.offsetWidth','HTMLCanvasElement.offsetHeight',
                 'window.screen.colorDepth', 'window.screen.pixelDepth'],
    'browser':  ['window.navigator.plugins', 'window.navigator.vendor', 'window.navigator.language',
                 'window.navigator.cookieEnabled', 'window.navigator.mimeTypes', 'window.navigator.userAgent'],
    'storage':  ['window.openDatabase', 'window.indexedDB', 'window.sessionStorage', 'window.localStorage'],
    'system':   ['window.navigator.maxTouchPoints', 'navigator.oscpu', 'navigator.hardwareConcurrency',
                 'navigator.cpuClass', 'navigator.platform'],
    'detectbot':['window.navigator.webdriver'],
    'store_fp': ['window.navigator.sendBeacon', 'window.fetch', 'XMLHttpRequest.send']
}
```

Figure 10: Diagram of Attributes used in Attribute-Based Scoring Heuristics

For each matched attribute, the total attribute count is incremented, and the corresponding bit position in the bitmask is set. The resulting score is expressed as a floating-point number with three decimal places in the format <d>.<ddd>, where d represents the count of matched attributes and <ddd> is the decimal representation of the bitmask. This encoding scheme offers a structured and quantifiable method for identifying specific fingerprinting behaviour, enabling both interpretability and computational efficiency in script analysis.

Additionally, each subcategory is associated with a decision threshold as defined in Table 6. These thresholds specify the minimum set of dynamically executed instructions required to infer the presence of a fingerprinting technique. For instance, a threshold score of 1.128 for audio fingerprinting subcategory signifies that the invocation of “OfflineAudioContext.oncomplete” event must be present and accompanied by the creation of additional audio configuration nodes to be classified as fingerprinting. By incorporating such contextual constraints, the framework enhances the classification reliability and reduces false positives, particularly in superficial or incidental API usage scenarios, such as the isolated invocation of “oncomplete”.

4.4.3 Joint-Attributes Heuristic Scoring

Following the individual attribute-based scoring, a joint-attributes heuristic is applied across all attribute scores to derive a final browser fingerprinting score. This composite score is used to access two types of behaviours: (a) invasive browser fingerprinting and (b) general fingerprinting.

```
def __analyse_fp_heuristics(self, attributes):
    cr_sym = {"audio" : 1.128, "canvas": 1.128, "font": 1.128, "webgl": 1.0, "webrtc": 1.128, "detectbot": 0.0}
    ncr_sym = {"screen": 0.0, "browser": 0.0, "storage": 0.0, "system": 0.0, "store_fp": 0.0}
    cr_score = 0
    ncr_score = 0
    ncr_attr_score = 0
    for cr in cr_sym.keys():
        if (float(attributes[cr]) > cr_sym[cr]):
            cr_score +=1

    for ncr in ncr_sym.keys():
        if(float(attributes[ncr]) > ncr_sym[ncr]):
            ncr_score +=1
            ncr_attr_score += int(attributes[ncr].split('.')[0])
    return f"{cr_score}.{ncr_score}{ncr_attr_score:02d}"
```

Table 6: Joint-Attributes Heuristic Scoring Threshold

The joint-attributes heuristic score is also expressed as a floating-point number with three decimal places in the format <d>.<d><dd>, where d represents a digit. The score comprises of three distinct components: (a) the count of matched critical fingerprinting subcategories, (b) the count of matched generic fingerprinting subcategories and (c) the total number of matched generic fingerprinting attributes. This encoding scheme provides a concise yet granular representation of the fingerprinting behavioural footprint exhibited by a given script. A higher joint-attributes heuristic score indicates a more intrusive level of invasive fingerprinting, thereby increasing the script's ability to uniquely distinguish users.

```
def __apply_bfp_decision(self, x, is_fpi: bool = True):
    if is_fpi:
        return (float(x['fpa_score']) > 2.0) or ((float(x['fpa_score']) >= 1.2) and (float(x['detectbot']) < 1.0))
    else:
        return (float(x['fpa_score']) >= 1.2)
```

Figure 11: Joint-Attributes Decision Threshold Algorithm

Finally, the browser fingerprinting decision threshold is applied to the joint-attributes heuristic score in accordance with the technical definitions defined in Section 4.4.1. The threshold mechanism enables the framework to classify scripts based on the cumulative behavioural footprint derived from both the critical and general fingerprinting groups. The implemented algorithm, which operationalises this classification logic, is illustrated in Figure 11.

4.5 Testing Methodology: Effectiveness of Chrome Browser Extension

This study sought to evaluate the effectiveness of heuristic-based browser extension defences in mitigating invasive browser fingerprinting, used for user identification, within the context of Google Chrome browsers. Specifically, it investigates whether there is a difference in the extension's ability to block fingerprinting scripts originating from first-party versus third-party domains. Additionally, the study examines the extent to which these extensions are effective against different levels of invasive fingerprinting, as measured by the joint-attributes heuristic score.

4.5.1 Selection and Configuration of Browser Selection

Google Chrome is selected as the test environment due to its dominant market share, making it a representative choice for real-world user behaviour. Despite its widespread adoption, Chrome is known to offer comparatively limited built-in privacy protections, thereby providing a better assessment for evaluating the efficacy of privacy-protecting extensions. The browser will be operated in its standard default configuration.

4.5.2 Selection and Configuration of Browser Extensions

Five MV3-compatible browser extensions are selected for evaluation based on the following criteria: (a) Citations in a minimum of three relevant literatures, (b) User rating of at least 4.5 and a user base exceeding 1M users, and (c) Demonstrated anti-tracking capabilities. The extensions that meet at least two

of the three criteria include AdGuard[5], Disconnect[6], Ghostery[7], Privacy Badger[8] and uBlock Origin Lite[9]. A detailed overview of the selection criteria and corresponding ratings for each extension is shown in Table 7.

Extension	Literature	Ratings / Users / Featured	Anti-Tracking
AdGuard	[14, 61, 65]	4.7 / 14M / Yes	✓
Disconnect	[43, 47, 53]	4.4 / 400K / Yes	✓
Ghostery	[11, 18, 38]	4.6 / 2M / Yes	✓
PrivacyBadger	[15, 25, 52]	4.4 / 1M / Yes	✓
uBlock Origin Lite	[49, 65, 66]	4.5 / 8M / Yes	✓

Table 7: Selection Ratings for the Five Selected Browser Extensions

All browser extensions are evaluated using their default configurations, specifically focusing on their built-in anti-tracking capabilities. In cases where anti-tracking features were not enabled by default, they are manually activated without any further customisation. Out of the five extensions, additional configuration is required for AdGuard and UBlock Origin Lite.

4.5.3 Selection of Test Cases

The test cases are derived from the results of the FPAnalyze dataset, which comprises (a) a mixture of first-party and third-party script domains, and (b) a balanced distribution of domains across the joint-attribute score range of 1 to 5. A total of 50 websites, including 20 first-party and 30 third-party script domains were selected and manually analysed to ensure sufficient coverage across the full spectrum. To avoid bias introduced by prior measurement study, the websites were sampled directly from the raw FPAnalyze dataset output, excluding any tracker category information.

The manual analysis involved validating the presence of specific fingerprinting behaviours identified by FPAnalyze. Upon validation, further examination is conducted to determine if the fingerprint was stored locally on the client or transmitted to a remote fingerprinting server. The locations of local storage and remote server are extracted and incorporated into the evaluation criteria defined in Section 4.5.4. Scripts identified as false positives or those that were heavily obfuscated are excluded from the test set to ensure verifiability and reproducibility of test results.

[5] https://chromewebstore.google.com/detail/adguard-adblocker/bgnkhhnnamicmpeenaelnjfhikgbkllg
[6] https://chromewebstore.google.com/detail/disconnect/jeoacafpbcihiomhlakheieifhpjdfeo
[7] https://chromewebstore.google.com/detail/ghostery-tracker-ad-block/mlomiejdfkolichcflejclcbmpeaniij
[8] https://chromewebstore.google.com/detail/privacy-badger/pkehgijcmpdhfbdbbnkijodmdjhbjlgp
[9] https://chromewebstore.google.com/detail/ublock-origin-lite/ddkjiahejlhfcafbddmgiahcphecmpfh

```
{
    "test_case": 14,
    "site_url": "http://facct.ru",
    "party": "1p",
    "script_url": "https://www.f6.ru/wp-content/themes/gib-theme/assets/bp.js",
    "fpa_score": "4.510",
    "site_rank": "11541",
    "fp_server": "https://f6.ru/api/fl",
    "fp_storage": ["gsscw-f6","fgsscw-f6"]
},
```


Figure 12: Sample Output of Manual Analysis

Finally, 15 first-party and 25 third-party domains were shortlisted as the final test set. A sample output of a test case derived from the manual analysis is shown in Figure 12. In this sample, the fingerprinting script “bp.js” was loaded. After execution, a fingerprint was generated and sent to the remote server using the URL “https://f6.ru/api/fl”. In addition, fingerprint related information was stored as cookies with name “gsscw-f6” and “fgsscw-f6”.

4.5.4 Criteria for Evaluation

Fingerprinting mitigation is defined as successful if it satisfies at least one of the following four criteria: (a) Direct blocking of fingerprinting script before loading, or (b) Indirect blocking, where the script is not requested for loading, presumably due to the extension blocking an associated upstream script, (c) Preventing storage of fingerprint as cookie or local storage data or (d) Prevention of remote transmission, whereby fingerprint data is not sent to external servers.

4.6 Test Environment Setup

The test environment is deployed on the same virtual machine setup and configuration in the OpenWPM measurement study. However, instead of OpenWPM, Selenium is deployed with the Chrome browser to perform automated browsing tasks. To avoid interference from multiple extension installations, each browser extension is installed in their own dedicated Chrome profiles.

Profiling of the test website is conducted in GUI mode at a screen resolution of 1920x1280, simulating typical user browsing behaviour. Prior to serving the test website, the chrome browser is launched and paused for 30 seconds to allow the browser extension to be fully loaded. Upon page load, cookie consent banners are programmatically accepted to enhance likelihood of full site functionalities, and a bot mitigation routine is executed to reduce the chances of automated client detection.

To ensure consistency across test cases, each site is allocated a maximum load time of 60 seconds, after which a timeout is enforced. Following the completion of each profiling session, network logs, stored cookies, and local storage artifacts are extracted and archived for post-processing, in accordance with the test evaluation criteria outlined in Section 4.5.4.

5 Implementation

This section introduces the detailed steps and procedures for the setting up, configuring and conducting of the entire research within a virtualised environment. The study is implemented using VMWare Workstation 17 Pro, a widely adopted virtualisation platform available under a free license. The VM is installed with an Ubuntu 20.04.6 LTS operating system, selected for its compatibility with the software tools essential for the study, in particular OpenWPM. The hardware configuration of the VM is also configured with support for audio and GPU rendering as detailed in Table 8.

Device	Configuration
Memory	8GB
Processors	2 processors and 1 core per processor
Hard Disk	80GB
Network Adapter	NAT
Sound Card	Auto detect. Connected and use default host sound card
Display	Auto detect. Accept 3D graphics up to 8GB

Table 8: Detailed Setup Configuration of VM

5.1 Web Measurement Study Using Instrumentation Tool OpenWPM

To support the deployment of OpenWPM, a conda environment (version 24.9.2) with Python version 3.12 is installed as a pre-requisite. This provides a consistent and isolated platform for managing software dependencies. Following this, the OpenWPM release package (version 0.31.0) is installed within the Ubuntu VM by executing the official installation script. The built-in unit-testing modules of OpenWPM are executed to verify that installation was successful and confirm that all its core components are functioning as intended. The framework is subsequently customised to extend its features and capabilities, tailoring it to meet the specific requirements set out in Section 4.2.1.

5.1.1 Accepting All Cookies

To streamline the handling of cookie consent prompts, OpenWPM's webdriver utility library is used to implement a custom command named "AcceptAllCommand". When initiated, the command waits for the web page to start loading before using XPath functionality to detect various implementations of consent banners. These implementations might include native control buttons or HTML elements, such as or , whose role attribute is explicitly defined as a button. Additionally, the label of the button must contain a term that matches at least one keyword from the predefined set of terms associated with cookie acceptance. The set of terms implemented in the keyword dictionary are: "Consent", "Accept" and "Allow". If a matching element is found within a default timeout period of 30 seconds, the command simulates a click to accept and dismiss the banner.

However, the command has two notable limitations: (a) non-English consent keywords are not supported, restricting its applicability to only English-language websites and (b) limited dictionary of predefined keywords, which may not cover

all possible phrasing variations in cookie labels across all websites and cookie consent managers. Examples of supported cookie banners are illustrated in Figure 13.

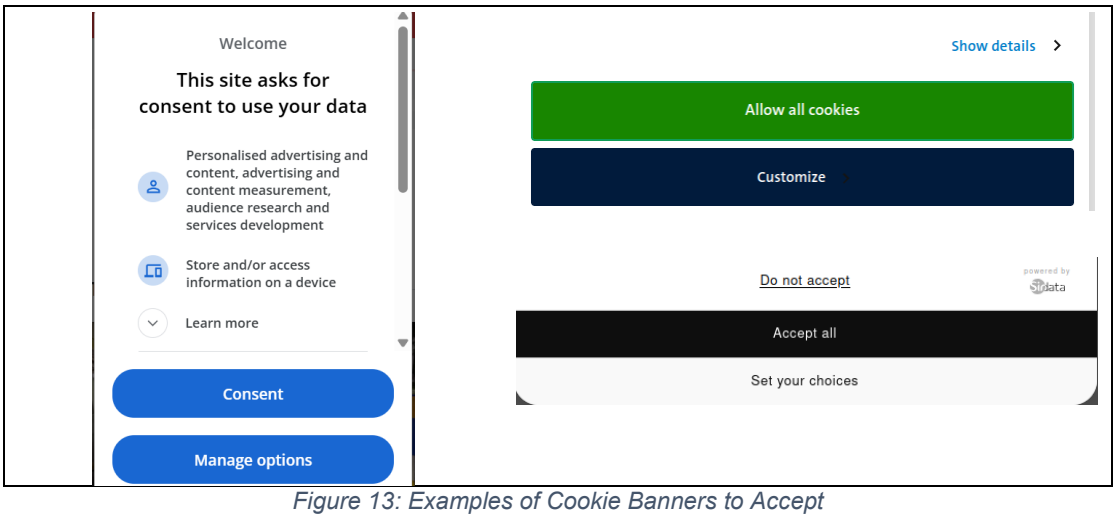

Figure 13: Examples of Cookie Banners to Accept

5.1.2 Profiling of Top 10,000 Tranco Websites

As outlined in Section 4.2, the study utilised the top one million most-visited websites from the Tranco [55] dataset, retrieved on 7th June 2025 with the unique identifier code VQ2VN[10]. The web crawling period was conducted between 25 Jun 2025 and 1 July 2025, with traffic routed through a VPN endpoint within the United Kingdom. This ensured consistency in regional content delivery during the data collection phase.

Due to constrained computing resources available and the anticipated large volume of data artifacts collected during the web profiling process; the web profiling script provided in OpenWPM is extended to conduct profiling in configurable cycles of 3000 – 4000 websites, with each cycle's artifacts segmented into different databases as shown in Figure 14. This modular approach aimed to facilitate efficient query execution across distributed and targeted databases, reducing resource contention and enhances flexibility in the organization and loading of databases for different use cases.

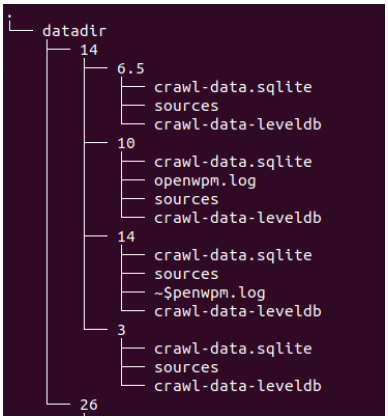

Figure 14: Diagram of Distributed Database Architecture

[10] Available at https://tranco-list.eu/list/ VQ2VN.

To accommodate for potential access failures arising from DNS resolutions or connectivity errors, the profiling scope was extended to include a total of 14,000 websites. This precautionary expansion ensures that even if a subset of sites proved unreachable, the resulting statistical analysis would remain robust and representative by mitigating the biases introduced by failed visits. The top 10,000 successful visits would be selected for further analysis.

5.1.3 Website Categorisation

To enrich the websites with site category information, a custom Selenium (version 4.27.1) implementation leveraging Chrome browser (version 138.0.7204.100) as an automated client was implemented to interface directly with Norton Safe Web via its query URL[11]. The implementation involves Selenium WebDriver dynamically loading the web content of the Norton Safe Web's queried result, and explicitly waiting for the appearance of a "div" element with class name "category-list" and label "CURRENT CATEGORY", as shown in Figure 15. Once the element is detected, the page source is parsed using the python library BeautifulSoup. The parser then iterates over all the span elements nested within "category-list" element, extracting and saving the list of associated category tags assigned to the domain.

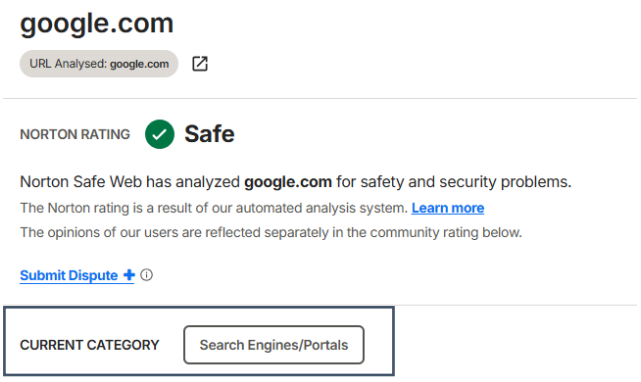

Figure 15: Snapshot of how Site Category is obtained from Norton Safe Web

5.2 Identifying and Contextualising Fingerprinting Websites

Unlike the Ghostery Tracking Protection Lists, which include both category mappings and an associated FilterList ruleset, the Disconnect datasets are limited to script domains identified as engaging in browser fingerprinting. To facilitate standardised and streamlined matching, domain entries of Disconnect are aggregated and translated into a compatible rule syntax. Specifically, domains are expressed using the format "||<domain>^", to allow the precise matching of the script's origin domain as shown in Figure 16. A detailed list of the versions of the resources used in this study is provided in Appendix A.2.

[11] https://safeweb.norton.com/report?url=<domain>

Figure 16: Sample of a Translated FilterList Rule Set for Disconnect

5.3 Implementation of FPAnalyze Heuristic Framework and Analysis

The FPAnalyze heuristic framework was developed in Python, leveraging data analytics libraries such as Pandas and Scikit-learn to efficiently process the large volume of data artifacts generated during profiling. To reduce the overhead associated with repeated SQL database queries, and improve access speed and search latency during analysis, all datasets retrieved and post-processed from each task were locally cached in JSON format as presented in Figure 17.

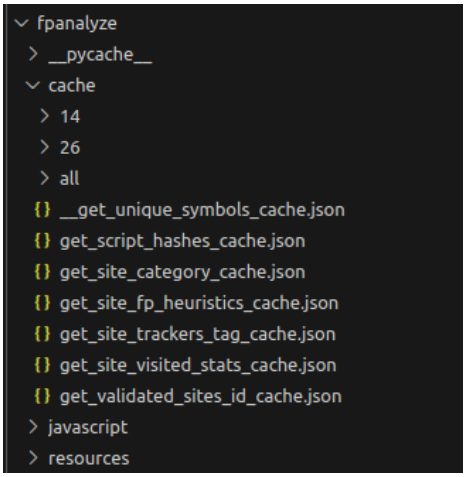

Figure 17: Snapshot of Cache Architecture For Individual Tasks

5.4 Testing Browser Extensions

A custom Python module named, FPBexTest, was developed to facilitate automated testing of browser extensions. This module leverages Selenium (version 4.27.1) and Google Chrome (version 138.0.7204.100) to setup the test environment, execute and manage test cases programmatically. The testing was conducted on 3 August 2025; with all browser traffic routed through a VPN endpoint located in the United Kingdom. This configuration ensured consistency in regional content delivery and browser environment setup when running each of the test cases. The detailed Google Chrome browser configuration for the test runs is outlined in Table 9.

Configuration Type	Parameter	Setting
Chrome Options	goog:loggingPrefs	Performance: ALL
	--disk-cache-size	0
	--ignore-certificate-errors	-
	--profile-directory	See Table 10

	--user-data-dir	custom
Selenium Driver	window size	1920 x 1080

Table 9: Selenium and Chrome Configuration

5.4.1 Setup of Chrome Browser Profiles

Prior to testing, individual profiles are created for each browser extension and configured according to the baseline settings listed in Table 10. Upon completion of the setup, the entire Google Chrome default user data directory[12] was archived to serve as both the baseline configuration and a backup of the browser environment. For each test case execution, the Chrome user data directory was reverted to its original state using this baseline, ensuring consistency across all test runs.

Profile Directory	Extension	Version	Configuration
Default	Ghostery	10.5.2	Default
Profile 1	uBlock Origin Lite	2025.804.1547	1. Enable Privacy Filter Lists: - AdGuard URL Tracking Protection - Block Outsider Intrusion into LAN
Profile 2	Privacy Badger	2025.5.30	Default
Profile 3	Disconnect	21.0.1	Default
Profile 4	AdGuard	5.1.127	1. Enable Privacy Filter 2. Enable Tracking Protection
Profile 5	-	-	Default Guest Profile

Table 10: Configuration Settings for Browser Extensions

5.4.2 Benchmark Test

A benchmark test comprising the 40 test cases was initially conducted using the control profile, designated as “Profile 5” in Table 10. This served as a reference point for evaluating subsequent results and verification that the target websites were functioning as expected without the interference from the browser extensions. The distribution of the test cases across the two test conditions are illustrated in Table 11, while detailed descriptions of each test cases are provided in Appendix C.1.

Script Domain Type	Degree of Invasiveness (Joint-Attributes Score)	No of Test Cases
First- Party	1	3
	2	4
	3	4
	4	4
	5	0
Third-Party	1	6
	2	6
	3	6
	4	6
	5	1
	Total	40

Table 11: Distribution of Browser Extension Test Cases

[12] ~/.config/google-chrome

5.4.3 Differentiating between Direct and Indirect Blocking from Network Logs

To determine whether a fingerprinting script URL or a fingerprint remote server URL is directly blocked by a browser extension, Google Chrome's network logs are analysed. As illustrated in Table 12, this involves correlating the requestId value between the Network.requestWillBeSent and Network.loadingFailed event messages for the target request URL. A block request is indicated by the presence of the value "net::ERR_BLOCKED_BY_CLIENT" in the errorText field of the Network.loadingFailed message, indicating that the browser extension has intercepted the request.

Conversely, indirect blocking of a script URL is inferred when the URL does not appear in any of the logged Network.requestWillBeSent event messages, indicating that the request was likely inactivated due to the potential blocking of an associated upstream parent script by the browser extension.

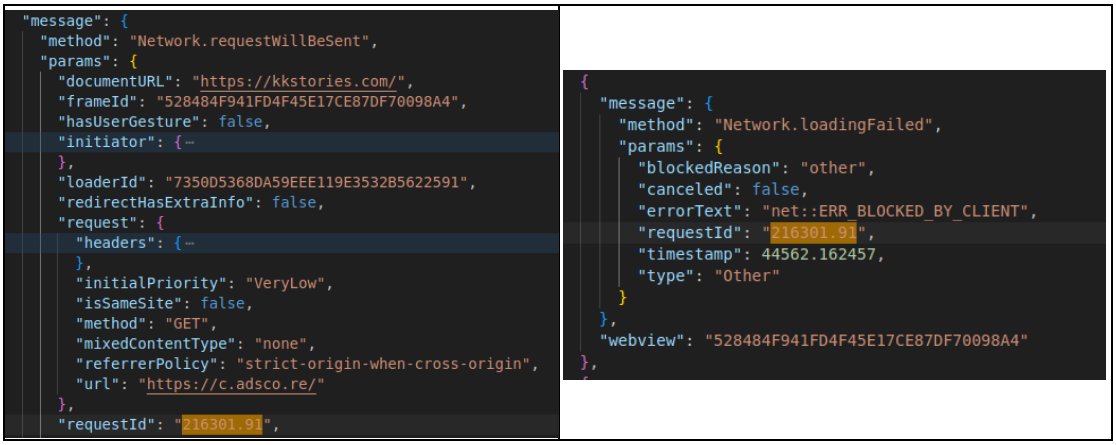

Table 12: Chrome's Network Logs Showing Script Blocked by Browser Extension

Table 13 represents a sample of the test results produced by FPBexTest. The result field captures the outcome for each of the three fingerprinting mitigation states: "block_script", "block_fp_server" and "block_fp_storage", indicating whether the browser extension successfully performed the intended blocking action. Each result was supplemented with relevant log entries, which serve as supporting evidence to substantiate the findings and reinforce the credibility of the evaluation.

Table 13: Sample Test Result Output

6 Results

This section presents the findings from the web profiling analysis and evaluation of the Chrome browser extensions.

6.1 Overview of Prevalence of Browser Fingerprinting

Out of the 14,000 sites profiled, a total of 11,374 sites were successfully visited. From this subset, the top 10,000 ranked sites were selected as the dataset for in-depth analysis. Detailed statistics pertaining to the profiling phase are provided in Appendix B. Following the categorisation of websites based on Norton Safe Web Classifications, Figure 18 presents the distribution of the top ten website categories. The three most frequently occurring website categories were Technology/Internet, News and Business/Economy.

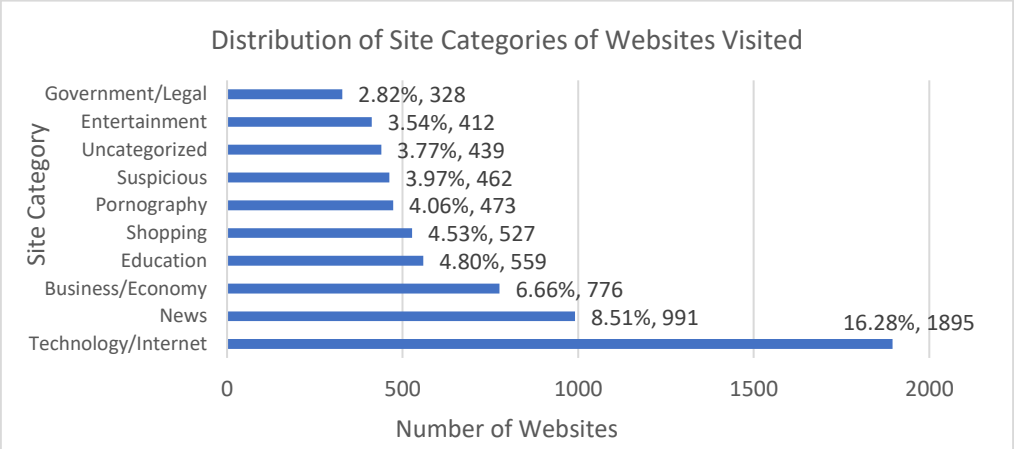

Figure 18: Distribution of top ten web site categorisation according to Norton Safe Web

The assessment of prevalence of browser fingerprinting was conducted using the Disconnect and Ghostery (D+G) TPL, which serve as the authoritative source of ground truth for this study. A total of 3855 websites (38.55%) were found to serve fingerprinting scripts, of which 606 websites (6.06%) employed invasive fingerprinting techniques for user identification. Figure 19 illustrates the raw counts of types of fingerprinting activities across different website categories. While the three most frequent categories involved in fingerprinting, Technology/Internet, News, and Business/Economy, are also the top three site categories of the profiled websites, notable differences emerged after normalising against the number of sites in each category as shown in Figure 20.

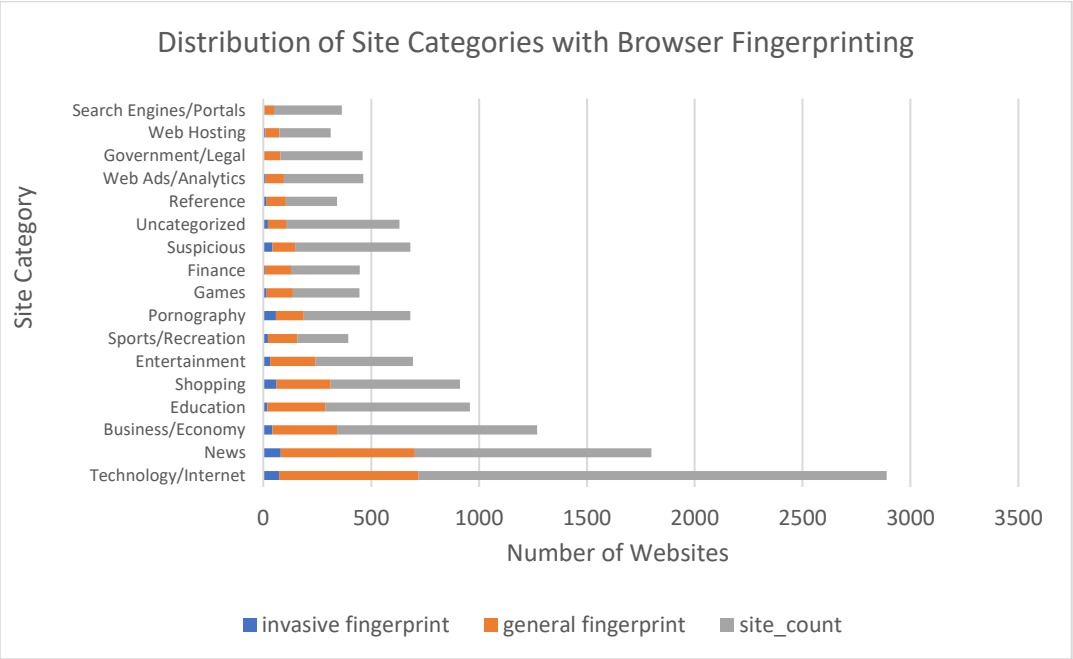

Figure 19: Distribution of top 10 Site Categories with Browser Fingerprinting

After normalising by category size, News, Education, Entertainment and Shopping websites remained within the top ten in terms of fingerprinting prevalence. However, fingerprinting occurred disproportionately higher in categories such as Sports/Recreational and Society/Daily Living categories, despite their smaller overall representation. Although Technology and Business/Economy were significantly dominant in overall site count, they exhibited relatively lower rates of fingerprinting when adjusted for their category size. Meanwhile, Shopping, Mixed Content/Potential Adult and Pornography emerged as the top three site categories for using invasive fingerprinting for user identification, at approximately 10-12% of websites of their own categories.

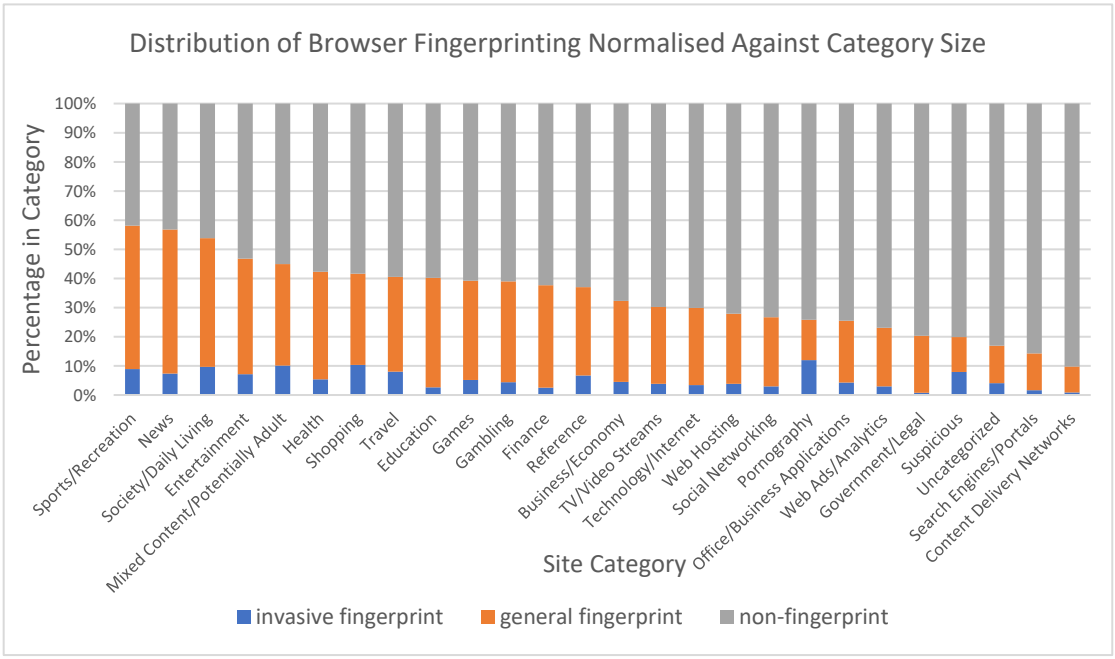

Figure 20: Distribution of Browser Fingerprinting Normalised Against Category Size

Among the 3855 unique websites identified as performing browser fingerprinting, over 97% were associated with third-party providers, distributed across 162 unique entities. Regardless of the kind of fingerprinting method used, the median number of third-party providers engaged in fingerprinting per affected site was one. These findings are illustrated in Figure 21.

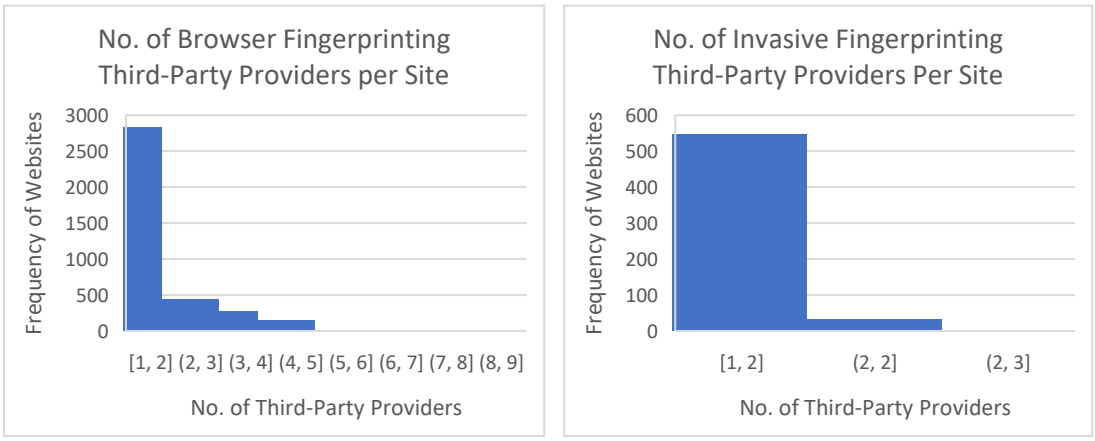

Figure 21: Charts of No. of Third-Pary Providers found on each Detected Website

Table 14 lists the top 10 ten third-party providers with a presence across multiple websites within the dataset. Notably, Google and Facebook are the only third-party entities appearing on more than 10% of the sites, indicating their prominence in the top-ranking sites. The majority of the top ten providers associated with invasive fingerprinting practices are largely providing advertising, bot detection and Content Delivery Network (CDN) services.

Browser Fingerprinting Third-Party Provider		**Invasive Fingerprinting Third-Party Provider**	
script_domain	**site_count**	**script_domain**	**site_count**
facebook.net	1811	**adsafeprotected.com**	83
google-analytics.com	1268	**bounceexchange.com**	50
doubleclick.net	1215	**alicdn.com**	29
googlesyndication.com	886	**flixcdn.com**	28
2mdn.net	262	**tapioni.com**	26
demdex.net	254	**px-cloud.net**	24
crwdcntrl.net	240	**clickcease.com**	20
fastclick.net	204	**wpadmngr.com**	20
adsrvr.org	192	**hcaptcha.com**	19
googleadservices.com	167	**flux-cdn.com**	19

Table 14: Top 10 third parties performing Browser Fingerprinting

6.2 Prevalence of Browser Fingerprinting in Digital Advertising

Among the top 10,000 websites analysed, 6,898 (68.98%) were identified by the D+G TPL as containing trackers linked to 933 distinct script domains associated with advertising services. Within this subset of advertising-associated websites, approximately 2,095 sites (~30%) exhibited browser fingerprinting behaviours, indicating a moderate prevalence level within the advertising ecosystems. In addition, a substantially smaller proportion of websites, 338 (4.89%), were observed to employ invasive fingerprinting for user identification.

Table 15 illustrates the distribution of different categories of browser fingerprinting trackers, along with the categorical proportion of those using invasive fingerprinting methods. The data reveals that invasive fingerprinting techniques, which are more capable of uniquely identifying users and persistently tracking users, are predominantly used within the advertising and anti-fraud domains. In contrast, less intrusive general fingerprinting techniques are likely to be more commonly used in social networking and web analytics categories. Interestingly, no anti-fraud trackers were identified to employ only general fingerprinting techniques.

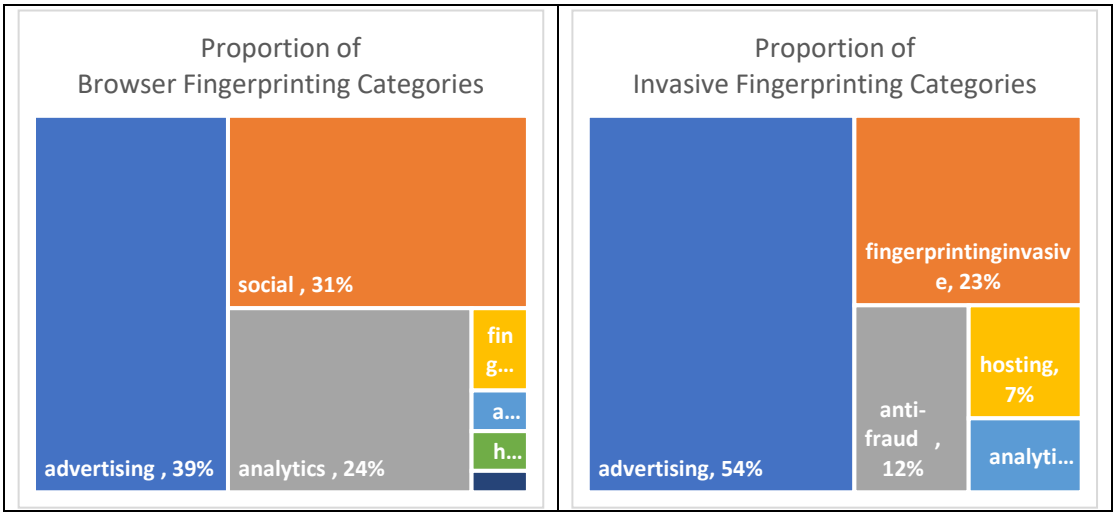

Table 15: Proportion of Tracker Categories associated with Browser Fingerprinting

6.3 Browser Fingerprinting Detection Performance of FPAnalyze

As part of the test case selection process, 50 JavaScript files identified by FPAnalyze as exhibiting invasive fingerprinting behaviours are manually analysed to validate the presence of fingerprinting techniques flagged by the tool's individual attribute-based scoring mechanism. During this manual analysis, 2 scripts are determined to be false positives, with FPAnalyze achieving an estimated precision rate of 96% in detecting invasive fingerprinting scripts.

After applying FPAnalyze to the measurement study dataset, a total of 3357 websites (33.57%) were found to serve fingerprinting scripts, of which 1240 websites (12.40%) employed invasive fingerprinting techniques for user identification. Among the 3357 websites identified as performing browser fingerprinting, 2734 (~81.44%) were associated with third-party providers, distributed across 546 unique entities.

Figure 18 presents a comparative analysis of the fingerprinting detection coverage across the two tools: D+G TPL and FPAnalyze. Each tool demonstrates distinct strengths. While FPAnalyze is less effective than D+G TPL in detecting general fingerprinting scripts, it performs better in identifying invasive fingerprinting websites. In addition, FPAnalyze outperforms D+G TPL in identifying invasive fingerprint scripts on first-party websites or their associated CDN services fronting their sites. Of the 170 unique fingerprinting

script domains detected by D+G TPL, 101 (~59.4%) domains were also identified by FPAnalyze, indicating a moderate overlap in detection capabilities.

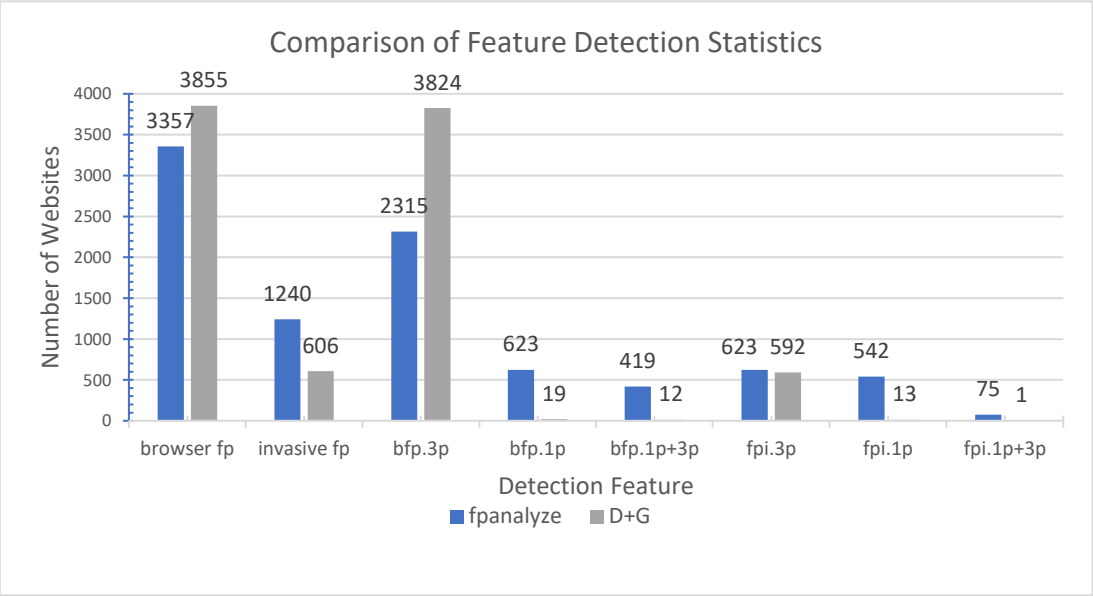

Figure 22: Comparative Analysis of Feature Detection Statistics

6.4 Attribute Collection Patterns in Browser Fingerprinting

The extent of fingerprinting attributes, as categorised in Section 4.4.1 is examined through a heatmap generated from the individual attribute-based heuristic scores. To visualise distinct usage patterns of fingerprinting attributes across scripts, a min max scaling procedure is applied to normalize all attribute scores to a range between 0 and 1. Subsequently, the mean of the normalised scored for each attribute is computed, serving as an indicator for its relative rate of occurrence. In the heatmap, darker shades indicate attributes that appear more frequently across fingerprinting scripts, thereby highlighting the more dominant techniques and subcategories.

With reference to Figure 23, a median general fingerprinting script uses at least 3 distinct subcategories within the general fingerprinting group, leading to the collection of approximately 6 unique attributes. As illustrated in the normalised attribute heatmap in Table 16, the most frequently targeted subcategories are the identification of supported storage mechanisms, such as open and indexed database, and system specifications, including max touch points, processor architecture and the operating system information. Notably, some of the characteristics associated to canvas, font, webGL and webRTC attributes were also collected in the process.

audio	canvas	font	webgl	webrtc	detectbot	screen	browser	storage	system	store_fp
0	0.004211	0.000463	0.013333	0.001429	0	0.414264	0.482786	0.617633	0.57119	0.211667

Table 16: Heatmap of Normalised Attributes used for General Fingerprinting

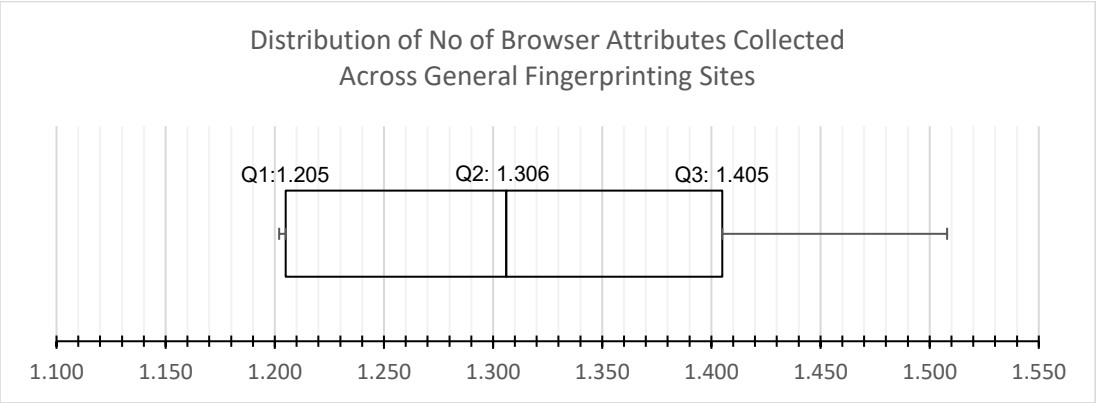

Figure 23: Distribution of Cluster of General Fingerprinting Attributes

A typical median invasive fingerprinting script collects at least two subcategories from the critical group and four from the general group. The small gap between the median and the 75th percentile suggests a narrow distribution in both the quantity and variety of attributes collected by invasive fingerprinting scripts. As illustrated in Figure 24, this reflects a consistent pattern in the nature and number of attribute collection across various invasive fingerprinting scripts. Based on the normalised attribute heatmap in Table 17, canvas and canvas font fingerprinting technique remain prevalent. Storage mechanisms and system information, similar to general fingerprinting scripts, are most likely to be always collected in tandem.

audio	canvas	font	webgl	webrtc	detectbot	screen	browser	storage	system	store_fp
0.341458	0.500659	0.464683	0.016775	0.119856	0.558935	0.544637	0.644174	0.797318	0.707858	0.045627

Table 17: Heatmap of Normalised Attributes used for Invasive Fingerprinting

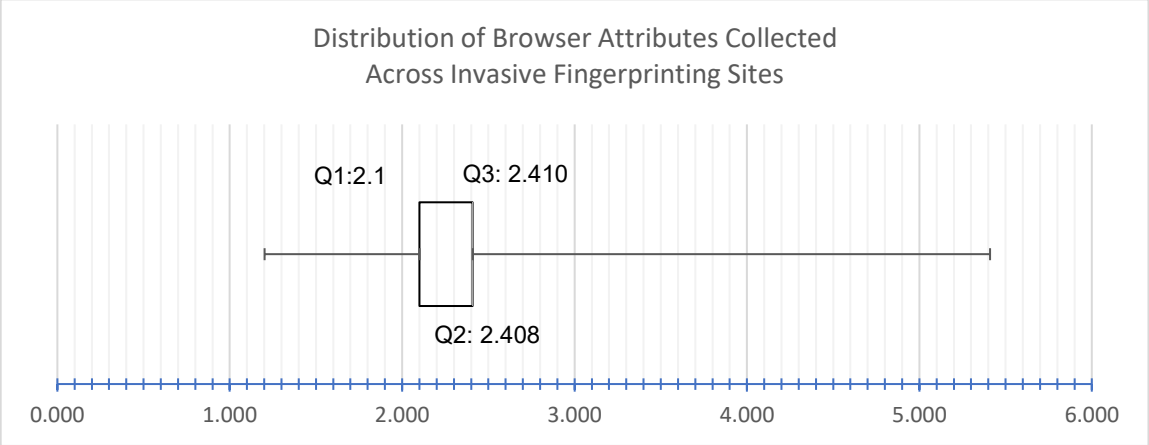

Figure 24: Distribution of Cluster of Invasive Fingerprinting Attributes

6.5 Evaluation of Browser Extension Defence

Table 18 presents the mitigation results for the five browser extensions across the 40 test cases. The percentage of successful mitigation ranges between 20% to 58%. Among them, Disconnect was the least-performing extension, mitigating only 20% of the identified browser fingerprinting scripts. In contrast, Ghostery and uBlock emerged as the most effective, exhibiting identical rates and blocking precisely the same script domains and URLs. When comparing their effectiveness against first-party and third-party script domains, both Privacy Badger and Disconnect failed to block any first-party domains.

Meanwhile, Ghostery and uBlock again achieved the highest mitigation rate, successfully blocking 40% of such domains.

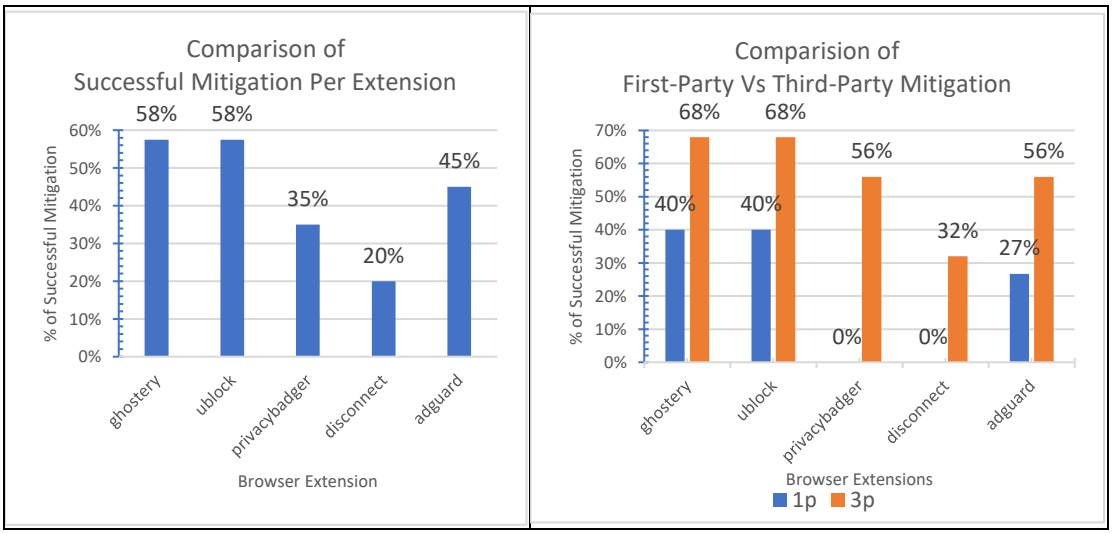

Table 18: Comparison of Performance of Browser Extensions

Figure 25 illustrates the mitigation rate across each band of FPAnalyze's Joint-Attributes Heuristic Score. With the exception of Disconnect, the mitigation performance of the browser extensions remains relatively stable across the score bands, with a standard deviation ranging from 0.14 to 0.19. In contrast, Disconnect exhibits a significant 50% drop in performance when the heuristic score exceeds 1. Notably, most extensions achieve their highest mitigation rate at heuristic scores of 1 and 3, while performance tends to be lowest when the score reaches 4.

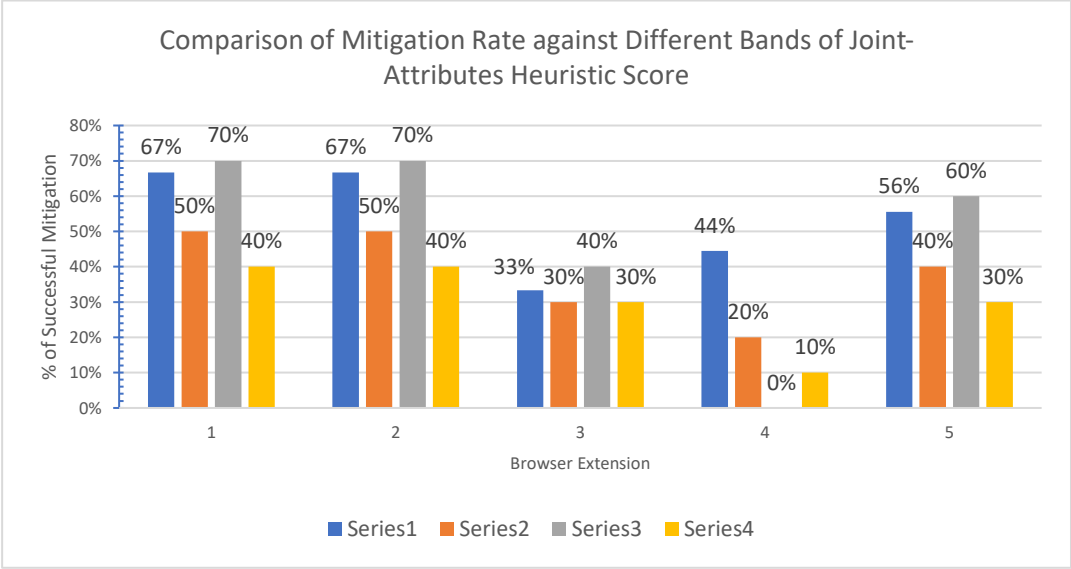

Figure 25:Comparison of Mitigation Rate across Range of Joint-Attributes Heuristic Score

7 Discussion

Based on the D+G TPL dataset, 38.55% of the top 10,000 most visited websites were observed to serve browser fingerprinting scripts, with 6.06% exhibiting invasive fingerprinting techniques capable of more precise user identification. These figures mark a notable rise from previous measurements, including the 22.76% recorded by FP-Inspector [4] in 2020 and the 33.58% observed by FProbe [9] in 2023, highlighting the growing prevalence of browser fingerprinting. Given the inherent limitations in D+G TPL's coverage, the actual extent of fingerprinting deployment is likely to be underrepresented.

7.1 Adoption of Browser Fingerprinting associated with Google Advertising Platforms

This upward trend may be partially driven by changes in Google's privacy policies, which were relaxed in February 2025 to allow usage of browser fingerprinting with its products. Supporting this hypothesis, manual static analysis of a fingerprinting script served from a news website, with SHA256 9d3480f9543601a96ab5482d3c56fa885fc02e22efdd6c6da243961feb16c48f, revealed the usage of invasive fingerprinting usage in conjunction with advertising services affiliated to Google. The script leverages the FingerprintJS library to generate a unique visitor identifier, "UDN_FID", and pushed it to Google Tag Manager (GTM) and Google Publisher Tag (GPT) present within the context of the page. The code snippet of the corresponding implementation is shown in Figure 26.

```
if (window.udnFingerprint.value = t.visitorId, localStorage.getItem("UDN_FID")) {
  const n = JSON.parse(localStorage.getItem("UDN_FID"));
  n.DFID = t.visitorId,
  localStorage.setItem("UDN_FID", JSON.stringify(n)),
  P.get("UDN_FID") || (console.log("rewrite exists to cookie"), P.set("UDN_FID", n.FFID, {expires: 365, domain:
} else {
  const n = {FFID: t.visitorId, DFID: t.visitorId, ISD: (new Date).toISOString()};
  localStorage.setItem("UDN_FID", JSON.stringify(n)), console.log("write cookie"), P.set("UDN_FID", t.visitorId
}
udnLayer.forEach(n => {
  n.type === "GTM" && (window.dataLayer = window.dataLayer || [], window.dataLayer.push({...n.data, ffid: JSON.
  n.type === "GPT" && (window.googletag = window.googletag || {}, window.googletag.cmd = window.googletag.cmd |
});
(), window.udnFingerprint = {};
```

Figure 26: Code snippet of integration of Fingerprinting with GTM and GPT

7.2 Relationship between Browser Fingerprinting and Site Categories

Prior studies [4, 38, 53] have explored the relationship between browser fingerprinting and site categories by measuring its prevalence as the proportion of site categories represented among the identified websites. By analysing the employment of fingerprinting scripts across various website categories, the present study corroborates earlier findings, suggesting that the use of fingerprinting is unevenly distributed across web site categories, with Technology/Internet (6.47%) and News websites (6.22%) exhibiting significantly higher levels of fingerprinting. In contrast, other categories such as Business/Economy (3%), which ranked third, showed substantially lower prevalence.

However, this approach may be biased due to the unequal distribution of website categories in the dataset, potentially overrepresenting categories with

a large number of sites. To address this limitation and enhance the validity of the findings, an alternative normalised metric, defined as the proportion of fingerprinting usage relative to the total number of websites within each category, was used. Additionally, web site categories comprising less than 1% of the total dataset are excluded from the analysis to mitigate the influence of outliers. Using this refined metric, the analysis reaffirmed that the News category continues to exhibit the highest level of fingerprinting activity. However, it also reveals that the level of fingerprinting across categories is more even than previously reported as illustrated in Figure 20. This observation suggests that while certain site categories are indeed more prone to fingerprinting activities, the disparity in fingerprinting levels may not be as pronounced than initially interpreted.

7.3 Prevalence of Browser Fingerprinting for User Identification in Digital Advertising

The findings indicate that browser fingerprinting is less predominantly employed for advertising purposes than initially anticipated, representing approximately 39% of the total identified instances. This proportion is comparable to that observed in the social and analytics categories, at 31% and 24% respectively. Even within the digital advertising ecosystem, the adoption rate also appears to be moderate, with only approximately 30% of advertising-related websites utilising fingerprinting. Furthermore, the use of invasive browser fingerprinting for user identification is relatively low, occurring in just 4.89% of those instances. These results suggest that while advertising might be one of the primary drivers of browser fingerprinting adoption, its role in comprehensive web tracking is less dominant compared to other tracking technologies, including first-party cookies, pixel tracking and universal identities. This observation is consistent with the findings of Martinez et al. [59], that tracking cookies and pixel tracking have the highest presence across websites.

Furthermore, the deployment of fingerprinting techniques appears to be context-dependent. Outside the advertising domain, general fingerprinting is predominantly used in social media and site analytics platforms, where the primary objectives include improving site performance and personalising content, often performed without requiring precise user identification. In contrast, invasive fingerprinting methods, which leverage high-entropy signals for robust identification, are more commonly adopted in security-focused applications such as bot detection and anti-fraud. This is likely due to a higher accuracy requirement to counter evasion techniques used by bots [65]. This pattern indicates a correlation between the required granularity of identification and the type of fingerprinting method adopted, suggesting that the industry does not exploit fingerprinting indiscriminately, but calibrate its use to align with specific contextual accuracy requirements.

7.4 Extent of Fingerprinting Attributes

The study provides further insights into the kinds of browser attributes collected by fingerprinting scripts. As illustrated in Section 6.4, a typical median general fingerprinting script gathers attributes from at least three of the five general subcategories, whereas invasive scripts span at least four. Furthermore, the study demonstrates the value of integrating bot detection-related attributes into

the broader fingerprinting detection framework. This is evident from the heatmap in Table 17, which reveals that bot detection attributes are more frequently captured than any invasive fingerprinting attribute. Notably, invasive scripts collect approximately twice as many attributes as their general counterparts, reflecting their need for more extensive data aggregation to achieve high-fidelity identification. These findings offer a more nuanced understanding of the behavioural distinctions between general and invasive browser fingerprinting methodologies.

7.5 Browser Extension as a Defence Mechanism

Among the five extensions evaluated, only Disconnect and uBlock Origin Lite mitigated more than 55% of the test cases, with both achieving a notably high mitigation rate of 68% against third-party service providers. Considering that the fingerprinting domains tested are largely absent from the top 10 most prominent invasive fingerprinting providers listed in Table 14, these results are particularly noteworthy and consistent with earlier findings from Englehardt et al. [38]. Furthermore, a 40% mitigation rate was observed for first-party providers, indicating a moderate level of protection. These outcomes aligned with the known limitations of heuristic-based protection mechanisms [4].

These extensions provide a reasonable level of protection against privacy-invasive fingerprinting. However, performance varied significantly across the evaluated extensions, highlighting the importance of careful browser extension selection to maximise privacy protection. Disconnect's comparatively lower performance may be attributed to its more stringent definition of trackers and its primary focus on third-party tracker [13], thereby limiting its overall effectiveness.

Given that only a relatively small percentage (6.06%) of the top 10,000 most visited websites use invasive fingerprinting for user identification, the overall threat to user privacy appears comparatively limited. In this context, the evaluated browser extensions can be deemed to be adequately effective. Nonetheless, the study acknowledged that the measurement study and test cases represent only a limited subset of the broader fingerprinting landscape and may not fully reflect the true limitations or capabilities of the browser extensions. To establish a more definitive conclusion, further research incorporating a more diverse and representative set of fingerprinting domains is warranted.

7.6 Limitations

The analysis presented in this study is subject to several limitations. Notably, it did not involve interactive engagement with websites beyond accepting cookie consent banners. This is in contrast to a more active browsing approach used in previous measurement studies from FP-Inspector [4] and OpenWPM [38]. As a result, the fingerprinting behaviours assessed are limited to those observed on landing pages, potentially excluding scripts that are triggered via further sub-

[13] https://disconnect.me/trackerprotection#definition_of_tracking

page navigation. This constraint potentially limits the comprehensiveness of the findings.

Moreover, the selection of JavaScript symbols instrumented for this measurement study could be more comprehensive. Manual analysis of the fingerprinting scripts revealed a broader range of attributes and APIs that could be instrumented to enhance coverage. These include data transmission APIs such as “window.fetch” and “XMLHTTPRequest”, as well as other attributes like media devices and timezone offset. Incorporating these elements into the instrumentation schema would allow for a more reliable detection of presence of fingerprinting behaviour.

7.7 Future Works

Although the measurement study of the top 10,000 websites offers a representative snapshot of browser fingerprinting practices, expanding the dataset to include the top 100,000 websites would yield a more accurate and comprehensive trend analysis. Additionally, the current passive profiling approach could be enhanced with more active engagement to trigger deeper script execution across sub-pages. The FPAnalyze module could also be extended to include more distinct general subcategories such as media devices and time zone to expand the general fingerprinting coverage.

8 Conclusion

The objective of this study is to examine the prevalence of browser fingerprinting as a method for user identification and assess its level of invasiveness within the context of digital advertising. This is a topic identified to be underexplored in prior literatures. To achieve this, a large-scale measurement study was conducted on the top 10,000 most visited websites from the Tranco list using the OpenWPM instrumentation platform. Tracker category mappings from two reputable privacy-focused companies were applied to identify and contextualise browser fingerprinting websites.

The study showed that 38.55% of the websites engaged in browser fingerprinting activities, marking a 5% increase as compared to the previous measurement conducted by Zhao [9] in 2023. This upward trend may be partially driven by relaxation in Google's privacy policies permitting its usage, as evidenced by a case study in which a news website pushed the generated fingerprint to Google Tag Manager (GTM) and Google Publisher Tag (GPT).

Despite this rise, fingerprinting was found to be less predominantly employed for advertising purposes than initially anticipated, accounting for approximately 39% of the total identified instances, comparable to the social media and site analytics categories. Notably, the adoption of invasive browser fingerprinting for user identification in digital advertising was considerably lower, observed in only 338 of the profiled websites. In addition, the deployment of invasive fingerprinting techniques appeared to be correlated to the level of identification required for a specific application use case. These patterns indicates that the industry does not exploit invasive fingerprinting indiscriminately, but rather calibrates its use for specific use cases such as bot detection and anti-fraud, where high-fidelity user identification is required to differentiate malicious bots from legitimate users. These findings suggest that privacy risks posed by invasive fingerprinting for user identification in digital advertising may be lower than previously anticipated.

To enhance detection and provide insights into the invasiveness of browser fingerprinting, a dynamic analysis-based heuristic framework named FPAnalyze was introduced. FPAnalyze not only achieved a precision rate of 96%, but also outperformed the measurement study in its capability to detect invasive fingerprinting, by an approximate factor of two. In addition, FPAnalyze offers a more nuanced understanding of the behavioural distinctions between general and invasive browser fingerprinting methodologies. Invasive fingerprinting not only targets more attribute groups, but also collects twice as many individual attributes as general fingerprinting, thereby improving accuracy in user identification. The novel approach of incorporating bot detection attributes into the broader browser fingerprinting schema was shown to be valuable, as these attributes were captured more frequently than any invasive technique observed.

To round off the study, an evaluation of the effectiveness of five popular browser extensions mitigating invasive fingerprinting was conducted. The results showed moderate effectiveness, with the best-performing extensions, Ghostery

and uBlock Origin Lite, mitigating just 58% of the test cases. Nonetheless, given the relatively limited prevalence of invasive fingerprinting for user identification, these extensions can be considered adequately effective in their privacy-protection capabilities.

However, the study acknowledges that the measurement and test cases represent only a limited subset of the broader fingerprinting landscape and may not fully capture the true threat levels or the capabilities of browser extensions. To establish more definitive conclusions, further research is warranted by incorporating a more diverse and representative set of fingerprinting domains, along with deeper analytical insights. Furthermore, FPAnalyze could be enhanced to incorporate more general subcategory groups such as media devices and time zone, thereby expanding its coverage of general fingerprinting techniques and improving detection accuracy.

9 Bibliography

1. *Digital advertising worldwide - statistics & facts*. 2024 August 16 [accessed 2025 March 24]; Available from: https://www.statista.com/topics/7666/internet-advertising-worldwide/#topicOverview.
2. Libert, T., *Exposing the hidden web: An analysis of third-party HTTP requests on 1 million websites.* arXiv preprint arXiv:1511.00619, 2015.
3. Castelluccia, C., et al., *Betrayed by Your Ads: Reconstructing User Profiles from Targeted Ads*. 2012, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 1-17.
4. Iqbal, U., S. Englehardt, and Z. Shafiq, *Fingerprinting the Fingerprinters: Learning to Detect Browser Fingerprinting Behaviors.* 2020.
5. Lin, M., et al., *Browsing without Third-Party Cookies: What Do You See?* 2024.
6. *Browser Market Share Worldwide 2024*. 2025 March 24 [accessed 2025 February]; Available from: https://gs.statcounter.com/browser-market-share/all/worldwide/2024.
7. *Privacy Sandbox for the Web*. 2025 March, 2025 [accessed 2025 March 21]; Available from: https://privacysandbox.com/intl/en_us/open-web/.
8. Newman, D., *Evaluating the effectiveness of defences to web tracking*. 2018.
9. Zhao, R., *Toward the flow-centric detection of browser fingerprinting.* Computers & security, 2024. **137**: p. 103642.
10. Nikiforakis, N., et al. *Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting*. 2013. IEEE.
11. Ukani, A., *Characterizing Browser Fingerprinting and its Mitigations.* 2023.
12. Almond, S. *Our response to Google's policy change on fingerprinting*. 2024 December 19, 2024 [accessed 2025 March 22]; Available from: https://ico.org.uk/about-the-ico/media-centre/news-and-blogs/2024/12/our-response-to-google-s-policy-change-on-fingerprinting/.
13. Google. *Manifest V2 support timeline*. 2025 [accessed 2025 Mar 31]; Available from: https://developer.chrome.com/docs/extensions/develop/migrate/mv2-deprecation-timeline.
14. Lukic, K. and L. Papadopoulos, *Privacy vs. Profit: The Impact of Google's Manifest Version 3 (MV3) Update on Ad Blocker Effectiveness.* arXiv preprint arXiv:2503.01000, 2025.
15. Al-Fannah, N.M., et al., *Beyond Cookie Monster Amnesia: Real World Persistent Online Tracking*. 2018, Springer International Publishing: Cham. p. 481-501.
16. Acar, G., et al. *FPDetective: dusting the web for fingerprinters*. 2013. New York, NY, USA: ACM.
17. Disconnect. *Our New Approach to Address the Rise of Fingerprinting*. 2020 [accessed 2025 Jul 22]; Available from:

https://blog.disconnect.me/our-new-approach-to-address-the-rise-of-fingerprinting/.
18. Laperdrix, P., et al., *Browser Fingerprinting: A Survey.* ACM transactions on the web, 2020. **14**(2): p. 1-33.
19. Vastel, A., et al. *FP-STALKER: Tracking Browser Fingerprint Evolutions*. in *IEEE Symposium on Security and Privacy*. 2018.
20. Lawall, A., *Fingerprinting and Tracing Shadows: The Development and Impact of Browser Fingerprinting on Digital Privacy.* 2024.
21. Sweeney, M. *User Identification*. 2020 [accessed 2025 Mar 1]; Available from: https://adtechbook.clearcode.cc/user-identification/.
22. Kapoor, K.K., Y.K. Dwivedi, and N.C. Piercy, *Pay-per-click advertising: A literature review.* The Marketing Review, 2016. **16**(2): p. 183-202.
23. Nikiforakis, N. and G. Acar, *Browse at your own risk.* IEEE spectrum, 2014. **51**(8): p. 30-35.
24. Sanchez-Rola, I., et al., *The web is watching you: A comprehensive review of web-tracking techniques and countermeasures.* Logic Journal of the IGPL, 2017. **25**(1): p. 18-29.
25. Al-Fannah, N.M. and C. Mitchell, *Too little too late: can we control browser fingerprinting?* Journal of Intellectual Capital, 2020. **21**(2): p. 165-180.
26. Ermakova, T., et al., *Web tracking-A literature review on the state of research.* 2018.
27. Bujlow, T., et al., *Web tracking: Mechanisms, implications, and defenses.* arXiv preprint arXiv:1507.07872, 2015.
28. Schmucker, N. *Web tracking*. in *SNET2 Seminar Paper-Summer Term*. 2011. Citeseer.
29. Dave, V., S. Guha, and Y. Zhang. *Viceroi: Catching click-spam in search ad networks*. in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 2013.
30. Jonker, H., et al., *Fingerprint Surface-Based Detection of Web Bot Detectors*. 2019, Springer International Publishing: Cham. p. 586-605.
31. Alaca, F. and P.C. Van Oorschot. *Device fingerprinting for augmenting web authentication: classification and analysis of methods*. in *Proceedings of the 32nd annual conference on computer security applications*. 2016.
32. Laperdrix, P., et al. *Morellian analysis for browsers: Making web authentication stronger with canvas fingerprinting*. in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. 2019. Springer.
33. Al-Fannah, N.M. *Making defeating captchas harder for bots*. in *2017 Computing Conference*. 2017. IEEE.
34. Zhang, D., et al., *A Survey of Browser Fingerprint Research and Application.* Wireless communications and mobile computing, 2022. **2022**: p. 1-14.
35. Sergey, M. *Audio Fingerprinting: What It Is + How It Works with Web API*. 2025 July 9,2025 [accessed 2025 Jul 20]; Available from: https://fingerprint.com/blog/audio-fingerprinting/.

36. Mowery, K. and H. Shacham. *Pixel perfect: Fingerprinting canvas in HTML5*. in *Proceedings of W2SP*. 2012.
37. Gómez-Boix, A., P. Laperdrix, and B. Baudry. *Hiding in the crowd: an analysis of the effectiveness of browser fingerprinting at large scale*. in *Proceedings of the 2018 world wide web conference*. 2018.
38. Englehardt, S. and A. Narayanan, *Online Tracking: A 1-million-site Measurement and Analysis.* 2016.
39. Chea, E. *Browser fingerprinting explained (+7 top techniques)*. 2024 [accessed 2025 March 22]; Available from: https://fingerprint.com/blog/browser-fingerprinting-techniques/.
40. Eckersley, P., M.J. Atallah, and N.J. Hopper. *How Unique Is Your Web Browser?* 2010. Berlin, Heidelberg: Springer Berlin Heidelberg.
41. Cao, Y., S. Li, and E. Wijmans. *(Cross-) browser fingerprinting via OS and hardware level features*. in *Proceedings 2017 Network and Distributed System Security Symposium*. 2017. Internet Society.
42. Reiter, A. and A. Marsalek. *WebRTC: your privacy is at risk*. in *Proceedings of the Symposium on Applied Computing*. 2017.
43. Salomatin, A.A., A.Y. Iskhakov, and R.V. Meshcheryakov, *Comparison of the Effectiveness of Countermeasures Against Tracking User Browser Fingerprints.* IFAC-PapersOnLine, 2022. **55**(9): p. 244-249.
44. Laperdrix, P., B. Baudry, and V. Mishra. *FPRandom: Randomizing core browser objects to break advanced device fingerprinting techniques*. in *International Symposium on Engineering Secure Software and Systems*. 2017. Springer.
45. Vastel, A., et al. *FP-Scanner: The privacy implications of browser fingerprint inconsistencies*. 2018.
46. Mostsevenko, S. *How We Bypassed Safari 17's Advanced Audio Fingerprinting Protection*. 2024 March 6 [accessed 2025 March 24]; Available from: https://fingerprint.com/blog/bypassing-safari-17-audio-fingerprinting-protection/.
47. Sim, K., H. Heo, and H. Cho, *Combating web tracking: analyzing web tracking technologies for user privacy.* Future Internet, 2024. **16**(10): p. 363.
48. Khattak, S., et al. *Do you see what I see? differential treatment of anonymous users*. 2016. Internet Society.
49. Drazner, C., et al., *Investigating the effectiveness of web adblockers.* arXiv preprint arXiv:1912.06176, 2019.
50. Snyder, P., A. Vastel, and B. Livshits, *Who filters the filters: Understanding the growth, usefulness and efficiency of crowdsourced ad blocking.* Proceedings of the ACM on Measurement and Analysis of Computing Systems, 2020. **4**(2): p. 1-24.
51. Mayer, J.R., *" Any person… a pamphleteer" Internet Anonymity in the Age of Web 2.0.* 2009.
52. Acar, G., et al. *The web never forgets: Persistent tracking mechanisms in the wild*. in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 2014.
53. Boussaha, S., et al., *FP-tracer: Fine-grained Browser Fingerprinting Detection via Taint-tracking and Entropy-based Thresholds.* Proceedings on Privacy Enhancing Technologies, 2024.

54. Li, T., et al. *FPFlow: Detect and Prevent Browser Fingerprinting with Dynamic Taint Analysis*. in *Cyber Security*. 2022. Singapore: Springer Nature Singapore.
55. Le, P., *Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation*, in *Proceedings of the 26th Annual Network and Distributed System Security Symposium , series = NDSS 2019 , year = 2019, month = feb, doi = 10.14722/ndss.2019.23386 ,*. 2019.
56. Amazon, *Wayback Machine - Alexa Static Top 1M*, 2023,[accessed Jul 13, 2025], Available from: https://web.archive.org/web/20230803120013/http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.
57. Englehardt, S. *Studies using OpenWPM*. 2025 Apr 21 [accessed 2025 Jun 10]; Available from: https://github.com/openwpm/studies/blob/main/studies.md.
58. Krumnow, B., H. Jonker, and S. Karsch, *Analysing and strengthening OpenWPM's reliability.* 2022.
59. Martínez, D., et al., *Large-scale web tracking and cookie compliance: Evaluating one million websites under GDPR with AI categorization.* Journal of network and computer applications, 2025. **242**: p. 104222.
60. Norton. *Safeweb*. 2024 [accessed 2025 Jul 12]; Available from: https://safeweb.norton.com/.
61. Dao, H. and K. Fukuda. *Alternative to third-party cookies: investigating persistent PII leakage-based web tracking*. in *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*. 2021.
62. Ghostery. *What Are Web Trackers? Online Tracker Categories*. 2025 [accessed 2025 Jul 23]; Available from: https://alb.ghostery.com/blog/how-ghostery-categorizes-trackers.
63. Utz, C., et al., *Privacy rarely considered: Exploring considerations in the adoption of third-party services by websites.* arXiv preprint arXiv:2203.11387, 2022.
64. Vastel, A., et al. *Fp-crawlers: studying the resilience of browser fingerprinting to block crawlers*. in *MADWeb'20-NDSS Workshop on Measurements, Attacks, and Defenses for the Web*. 2020.
65. Venugopalan, H., et al., *FP-Inconsistent: Detecting Evasive Bots using Browser Fingerprint Inconsistencies.* 2024.
66. Senol, A., et al., *The Double Edged Sword: Identifying Authentication Pages and their Fingerprinting Behavior.* 2024.

Appendices

Appendix A.1

```
[
  {"ScriptProcessorNode": { "logCallStack": true }},
  {"GainNode": { "logCallStack": true }},
  {"AnalyserNode": { "logCallStack": true }},
  {"OscillatorNode": { "logCallStack": true}},
  {"OfflineAudioContext": { "logCallStack": true }},
  {"AudioContext": { "logCallStack": true}},
  {"RTCPeerConnection": { "logCallStack": true }},
  {"HTMLCanvasElement": { "logCallStack": true }},
  {"Storage": { "logCallStack": true }},
  {"window.navigator":{ "logCallStack": true }},
  {"CanvasRenderingContext2D": { "logCallStack": true }},
  {"window": { "propertiesToInstrument": ["name", "localStorage", "sessionStorage"], "logCallStack": true }},
  {"window.document": { "propertiesToInstrument": ["cookie", "referrer", "createComment", "createElement"], "logCallStack": true }},
  {"window.screen": { "propertiesToInstrument": ["pixelDepth", "colorDepth"], "logCallStack": true}},
  {"window.Animation": { "logCallStack": true }},
  {"window.EventTarget": { "propertiesToInstrument": ["addEventListener"], "logCallStack": true }},
  {"window.performance": { "logCallStack": true }},
  {"window.WebGLRenderingContext": { "logCallStack": true}},
  {"window.WebGL2RenderingContext": { "logCallStack": true}},
  {"window.WebGLActiveInfo": { "logCallStack": true }},
  {"window.WebGLBuffer": { "logCallStack": true }},
  {"window.WebGLContextEvent": { "logCallStack": true }},
  {"window.WebGLFramebuffer": { "logCallStack": true }},
  {"window.WebGLProgram": { "logCallStack": true }},
  {"window.WebGLQuery": { "logCallStack": true }},
  {"window.WebGLRenderbuffer": { "logCallStack": true }},
  {"window.WebGLSampler": { "logCallStack": true }},
  {"window.WebGLShader": { "logCallStack": true }},
  {"window.WebGLShaderPrecisionFormat": { "logCallStack": true }},
  {"window.WebGLSync": { "logCallStack": true }},
  {"window.WebGLTexture": { "logCallStack": true }},
  {"window.WebGLTransformFeedback": { "logCallStack": true }},
  {"window.WebGLUniformLocation": { "logCallStack": true }},
  {"window.WebGLVertexArrayObject": { "logCallStack": true }}
]
```

Figure 27: JS Instrumentation Configuration as specified in Section 4.2.1

Appendix A.2

Resource	Url
Disconnect Category TPL	https://raw.githubusercontent.com/disconnectme/disconnect-tracking-protection/735f1ae8cea77e4042c03cf83be3e03ead38d61b/services-relay.json
Ghostery Category TPL	https://github.com/ghostery/trackerdb/releases/download/20250710143 6/trackerdb.json
Ghostery FilterList	https://github.com/ghostery/trackerdb/releases/download/20250710143 6/trackerdb.txt"
Ghostery adblocker Engine Version	https://registry.npmjs.org/@ghostery/adblocker/-/adblocker-2.11.1.tgz
Alexa Top 1 million archive	https://web.archive.org/web/20230803120013/http://s3.amazonaws.com /alexa-static/top-1m.csv.zip
Norton SafeWeb	https://safeweb.norton.com/report?url=<domain>

Table 19: Detailed versions of resources as specified in Section 5.2

Appendix B.1

command_status	count
ok	10301
neterror	2624
timeout	1073
error	2
Total	14000

Table 20: Tranco Top 14K OpenWPM Crawl Status specified in Section 6.1

neterror	count
dnsNotFound	2069
connectionFailure	456
netReset	77
nssFailure2	12
redirectLoop	10
Total	2624

Table 21: Distribution of Network Errors Encountered as specified in Section 6.1

Appendix C.1

test_case	site_url	party	fpa_score	fp_server	fp_storage
1	http://mail.ru	1p	1.206	y	
2	http://blitz.gg	1p	1.205		
3	http://sport-fm.gr	1p	1.204		
4	http://bc.co	1p	2.304		
5	http://nola.com	1p	2.409		
6	http://innovid.com	1p	2.408	y	
7	http://adobe.com	1p	2.410	Y	
8	http://rambler.ru	1p	3.306		
9	http://sbis.ru	1p	3.407		
10	http://doublelist.com	1p	3.407		
11	http://goldapple.ru	1p	3.510	Y	y
12	http://businessnewsdaily.com	1p	4.408	Y	
13	http://coupert.com	1p	4.410		
14	http://facct.ru	1p	4.510	y	y
15	http://getjobber.com	1p	4.409		
16	http://jjshouse.com	3p	1.202	y	
17	http://comcast.net	3p	1.205	y	
18	http://dzen.ru	3p	1.206	y	
19	http://upwork.com	3p	1.236		
20	http://bilibili.com	3p	1.203		
21	http://crateandbarrel.com	3p	1.511	y	
22	http://simply.com	3p	2.000	Y	
23	http://livejasmin.com	3p	2.101		
24	http://rambler.ru	3p	2.305		y
25	http://udn.com	3p	2.407		
26	http://sattasport.in	3p	2.408		
27	http://timeweb.ru	3p	2.409		
28	http://swrve.com	3p	3.407		
29	http://dafiti.com.br	3p	3.407		
30	http://latamairlines.com	3p	3.407		
31	http://besoccer.com	3p	3.409		
32	http://webfx.com	3p	3.510	y	
33	http://bokeptoket.com	3p	3.510	y	
34	http://yougov.com	3p	4.307		
35	http://statefarm.com	3p	4.408		
36	http://espressif.com	3p	4.410		
37	http://snackvideo.com	3p	4.411		
38	http://lightinthebox.com	3p	4.508		
39	http://kkstories.com	3p	4.512		
40	http://viator.com	3p	5.410		
				13	**3**

Table 22: Detailed Test Cases Websites specified in Section 5.4.2

Appendix D.1

1. "Check for grammatical and spelling errors”