Comparative Privacy Analysis of Mobile Browsers

Ahsan Zafar
azafar2@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Anupam Das
anupam.das@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

ABSTRACT

Online trackers are invasive as they track our digital footprints, many of which are sensitive in nature, and when aggregated over time, they can help infer intricate details about our lifestyles and habits. Although much research has been conducted to understand the effectiveness of existing countermeasures for the desktop platform, little is known about how mobile browsers have evolved to handle online trackers. With mobile devices now generating more web traffic than their desktop counterparts, we fill this research gap through a large-scale comparative analysis of mobile web browsers. We crawl 10K valid websites from the Tranco list on real mobile devices. Our data collection process covers both popular generic browsers (e.g., Chrome, Firefox, and Safari) as well as privacy-focused browsers (e.g., Brave, Duck Duck Go, and Firefox-Focus). We use dynamic analysis of runtime execution traces and static analysis of source codes to highlight the tracking behavior of invasive fingerprinters. We also find evidence of tailored content being served to different browsers. In particular, we note that Firefox Focus sees altered script code, whereas Brave and Duck Duck Go have highly similar content. To test the privacy protection of browsers, we measure the responses of each browser in blocking trackers and advertisers and note the strengths and weaknesses of privacy browsers. To establish ground truth, we use well-known block lists, including EasyList, EasyPrivacy, Disconnect and WhoTracksMe and find that Brave generally blocks the highest number of content that should be blocked as per these lists. Focus performs better against social trackers, and Duck Duck Go restricts third-party trackers that perform email-based tracking.

CCS CONCEPTS

• **Security and privacy → Browser security**.

KEYWORDS

Web tracking; User privacy; Mobile browsers

ACM Reference Format:
Ahsan Zafar and Anupam Das. 2023. Comparative Privacy Analysis of Mobile Browsers. In *Proceedings of the Thirteenth ACM Conference on Data and Application Security and Privacy (CODASPY '23), April 24–26, 2023, Charlotte, NC, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3577923.3583638

CODASPY '23, April 24–26, 2023, Charlotte, NC, USA

ACM ISBN 979-8-4007-0067-5/23/04…$15.00
https://doi.org/10.1145/3577923.3583638

1 INTRODUCTION

We spend a significant portion of our daily routine surfing the web. Today, we interact with the web in almost every facet of our lives: socializing, banking, health care, education, and entertainment. As a result, over the years, we have seen a steady increase in online tracking activities across websites [55]. With the rapid adoption of smartphones, mobile devices as of 2016 have started generating more web traffic than their desktop counterparts [36, 46]. This has also prompted the development of new mobile-friendly web browsers as well as privacy-focused versions.

While numerous empirical studies have been conducted to understand the prevalence of online tracking in the wild [33, 34, 42, 43, 47, 55, 60], most of such analysis has been done in the context of a desktop browser. Furthermore, app-based mobile tracking has received some attention [62, 64], but not much research has been done to analyze and compare mobile browsers, especially privacy-focused mobile browsers.

However, the mobile platform presents multiple distinct characteristics compared to the desktop platform. For example, mobile platforms enable additional avenues to track users due to the presence of several embedded sensors that are directly accessible to JavaScript programs, often accessible without any user permission. Das et al. [40] have shown that scripts in the wild have started exploiting motion sensors to create unique device fingerprints. Moreover, the content served to mobile browsers can differ greatly from those served on their desktop counterparts. Yang et al. [66] showcased that almost 26% of the Alexa top one million websites serve mobile-specific pages with significant structural differences from their desktop counterparts.

In order to close the gap and better understand how popular mobile browsers fare against each other in terms of their effectiveness against online tracking, we present a large-scale study on both popular mobile browsers (e.g., Google Chrome, Firefox, and Safari) as well as privacy-focused browsers (e.g., Brave, Duck Duck Go, Firefox-Focus). The listed mobile browsers were chosen based on their popularity on the platforms' application stores [26]. We use data-driven analysis to understand the implications of browser choice for a user and answer the following research questions — **RQ1: *What are the major differences in the web traffic generated for different mobile browsers?*** We first analyze the distribution of different types of content encountered across different mobile browsers. Given cookies are an important mode of stateful tracking, we next analyze the distribution of cookies per website. To see the differences in API usage, we also look at the counts of web APIs and property accesses made by each browser. We divide the APIs and property accesses into general categories to better summarize the observed web-behavior. Leveraging the execution trace of scripts that we record in our measurement, we then observe how many times each class of API was called in a browser.

Our analysis helps us understand the major differences that arose in our crawls. **RQ2: *How effective are mobile browsers in blocking privacy-intrusive web content?*** We use popular blocking lists to provide a comparative analysis of how mobile browsers are blocking different web content. We assume that the closer a browser emulates the ad and tracker blocking list, the higher its efficacy is in terms of privacy protection. The goal is to evaluate the effectiveness of different privacy-focused mobile browsers in terms of how well they block well-known tracking content. We measure the should-be-blocked domains and observe the extent of tracking as seen in each browser. **RQ3: *Do invasive fingerprinters discriminate against users based on the browser they use?*** We define invasive fingerprinters as scripts that use non-traditional fingerprinting heuristics to target browsers. Using dynamic analysis, we apply state-of-the-art heuristics to first identify such opportunistic fingerprinters and then observe their behavior across browsers. Next, we reveal the number of websites that were targeted by such scripts across each browser. Then we explain why certain browsers may guard better against such tracking mechanisms. We also use static analysis to measure content-level differences that may arise as script authors may deterministically serve altered versions of the code to users based on their browser choice.

To answer these research questions, we first collect data using real Android and iPhone devices. Next, we collected data from six popular mobile browsers, including three privacy-focused browsers. To ensure fair comparison across all the browsers, we successfully load the same 10K websites from the Tranco 1-million websites [54] across all browsers. Using `mitmproxy` as our HTTP proxy, we also inject scripts to track the runtime access to well-known privacy-invasive web APIs and DOM properties. This enables us to dynamically analyze web behavior encountered across different mobile browsers. We also perform static analysis on the common JavaScript codes encountered across all browsers to highlight differential behavior across different mobile browsers. Lastly, we replicate our analyses across two datasets, collected at two different timestamps, to showcase the consistency of our findings.

In this paper, we make the following contributions:

- We perform a large-scale comparative analysis of mobile web browsers by crawling the Tranco 10K+ websites using six popular mobile browsers: Chrome, Firefox, Firefox Focus, Duck Duck Go, and Brave. We develop an automated crawling system using real smartphones to collect the necessary data, including source codes and execution traces of the scripts. We have publicly released our code and dataset. [1]
- We conduct a differential analysis to compare and contrast the traffic-level characteristics, as well as the contents served to different browsers. In addition, we observe the distribution of first and third-party requests and cookies set among the browsers. We also analyze the different APIs accessed across the six mobile browsers.
- For different privacy-focused browsers, we analyze the effectiveness of blocking unwanted web content using popular blocking lists, including EasyList, EasyPrivacy, Disconnect, and WhoTracksMe. Our results show that Brave generally outperforms other browsers in terms of protecting privacy-invasive content.
- Lastly, we perform static and dynamic analysis using source code and execution traces of the code to observe any differential treatment by the scripts and websites on the browsers.

The remainder of the paper is organized as follows. Section 2 describes the related work. Section 3 discusses our data collection methodology. Section 4 investigates the content-wise differences that arise when different browsers visit a particular website. Section 5 evaluates the effectiveness of the mobile browsers against well-known filter lists. Section 6 investigates the differential treatment by the browsers' scripts and websites. Section 8 summarizes the findings and lists the limitations of our approach. We conclude in Section 9.

2 RELATED WORKS

Online tracking imposes privacy risks for online users. In this section, we will review the literature involving different web tracking techniques across online services. We will also review different state-of-the-art defense strategies and measurement studies.

Web Tracking Techniques. Online tracking enables various trackers to collect users' browsing activities for different types of behavioral analysis, targeted advertisements, and surveillance [52, 63, 65].

Traditional web tracking or stateful tracking uses HTTP cookies as a dominant technique and also other conventional mechanisms, e.g., HTML5 storage [44], Flash cookies [35], Etags [33]. On the other hand, advanced stateless tracking techniques such as browser or device fingerprinting are used for identifying users with a specific browser state [49, 53, 60].

Web Tracking Measurement. Many empirical studies have looked at understanding the current trends of web tracking, specifically for desktop-based tracking. Krishnamurthy et al. [52] presented a longitudinal measurement study of web tracking, examining the prevalence of third-party trackers on the web. Mayer et al. [58] provide an analysis regarding the online-tracking strategies followed by different parties. To measure third-party tracking, they also introduce a web measurement platform, ForthParty [57]. Other studies have also identified new fingerprinting scripts and analyzed the prevalence of such web tracking techniques in the wild (e.g., FPDetective [34] and OpenWPM [43]). Other studies have shown that at least one third-party tracker monitors 46% of the 10k most popular websites (Alexa ranking), and one-third of the third-party requests are sent to tracker [37, 56]. Through an extensive study on Internet archive data from 1996 to 2016, Lerner et al. [55] suggested that web tracking has become more sophisticated and needs more attention. Studies have also proposed models for measuring browser fingerprinting based on certain characteristics of browser attributes which have been classified according to their prominence and volatility [49].

Mobile Web Tracking. Mobile web browsing has steadily grown since 2009, especially as mobile device usage overtook desktop and increasing availability of mobile-friendly websites [46]. Although mobile is now generating more web traffic than its desktop counterpart and incurs more sensitive information, little is known about web-tracking in mobile environment [45]. Yang et al. [66]

[1] https://github.ncsu.edu/azafar2/mobile_browser_privacy_analysis.git

have compared the mobile and desktop versions of 23,310 websites. They found that mobile tracking has unique characteristics due to mobile-specific trackers and is increasingly becoming prevalent as its desktop counterpart. A similar outcome was demonstrated by Das et al. [40]. Kondracki et al. [51] have shown the possible privacy and security threats of Android's data-saving browsers. Researchers have also studied the security and privacy risk in terms of in-app advertisement for both paid and free applications on mobile devices [48, 61, 64]. In addition, mobile browsers can potentially access hardware typically unavailable on desktops, like motion sensors, which can be used for tracking. Cassel et al. [38] performed a web measurement study comparing desktop browsers with their mobile variants to highlight these different tracking mechanisms. Our study is different from their work in that we specifically compare the strengths and weaknesses of generic and privacy browsers among each other on the same platform (i.e., mobile platform).

Distinction with Prior Work. In this paper, we perform a comparative study on mobile browsers — covering both popular browsers (e.g., Chrome, Firefox, and Safari) and privacy-focused browsers (e.g., Brave, Duck Duck Go, Firefox Focus). Through our analysis, we shed light on how different contents are filtered across different browsers and how different third-party domains serve discriminative scripts based on the type of browser requesting such scripts.

3 DATA COLLECTION AND PRE-PROCESSING

In this section, we describe our data collection setup and the different datasets we analyze in the rest of the paper.

Data Collection Setup. Our data collection setup consists of five Android mobile devices (Google Pixel 4a), one iPhone 11, and a desktop machine. Using the Android Debug Bridge (`adb`) [6], we automated the crawling process on the Android mobile device for a specific browser. The crawling script opens a browsing app on the mobile device and visits top-ranked sites listed by Tranco [54]. We use Appium [20] for iPhone automation and hook the client to the Safari web agent. The traffic data is sent to a proxy server that is hosted on a remote machine. We instantiate `mitmproxy` [16] server to intercept HTTP requests and log details, including the request URL, referrer, content type, content length, and response code. To capture dynamic runtime execution, we use the approach defined by OmniCrawl [38] and inject a script at the head of the website document file that logs execution of API calls along with script name and character offset within that script where the call was made. The web crawling script runs from a desktop, to which all the mobile devices are connected.[2] To handle HTTPS traffic, we installed `mitmproxy` CA certificate on the phone to bypass SSL pinning. The timeout period was set to 30 seconds for each site, and for any page-load failure, we made two retry attempts with a delay of 5 seconds between consecutive attempts. After each site visit, we cleared all data (e.g., cookies) associated with the browser app. Figure 1 highlights our data collection setup.

Data Sets. We collected data using six different browsers: Firefox, Chrome, Safari, Firefox-Focus [13], Brave [8], and Duck Duck Go [10]. Our datasets, therefore, not only contain data from general

[2]Both the mobile devices and the desktop machine were connected to the same WiFi network.

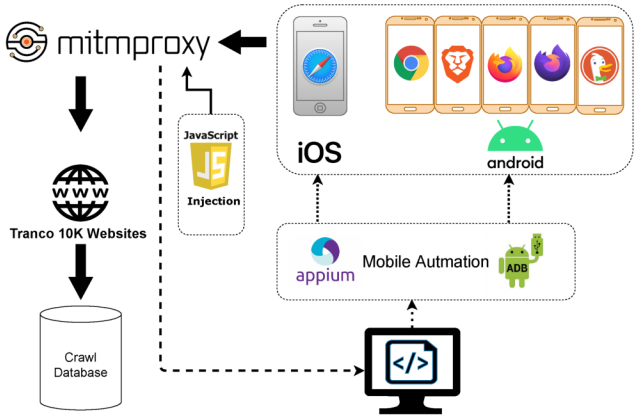


Figure 1: Data collection setup. We collect data using six different browsers on real-world mobile devices. We use `mitmproxy` to intercept HTTP requests and log details, including the request URL, referrer, content type, content length, and response code.

browsers like Chrome, Safari, and Firefox but also privacy-focused browsers like Brave, Firefox-Focus, and Duck Duck Go. The following browser versions were used throughout the study: Chrome (83), Firefox (96), Firefox Focus (95), Safari (15.4.1), Brave (1.34), Duck Duck Go (5.107). While many of the browsers use the same rendering engine like Blink [7] or Gecko [14] under the hood, the way each browser implements its tracker and ad-blocking techniques is different and, at times, proprietary. Past works have also used user-emulating web surfing tools like OpenWPM [15] to crawl websites. However, Demir et al. [41] have demonstrated that changing the environment of the web agent can induce variability in the web interactions encountered. This results in non-representative data sets that fail to capture organic user behavior. Hence, it is critical to obtain data using real browsers for ecological validity when comparing the effectiveness of different browsers [38].

Moreover, website contents typically change over time; we, therefore, collect two rounds of data approximately one month apart (the first round collected in June 2022 and the second in July 2022). We launched all browser crawls simultaneously to reduce the impact of variable content distribution due to dynamic content loading.

Data Preprocessing. While crawling the websites, we collected all the additional requests made through the browsers. In some cases, our automated script for visiting websites failed to successfully load the websites even in two attempts because of request timeout or network issues. So to conduct a fair differential analysis among the browsers, we only examined the sites successfully loaded across *all* six browsers. Many non-public-facing domains, such as CDNs and DNS (e.g., *akadns.net*), are present in the Tranco List. A user cannot trivially crawl these domains. Therefore, we adopted a slightly different approach to gathering data to mitigate the high failure rate when visiting the top 10K websites and ensure a fair comparison across all six browsers. We continue loading websites using the Tranco top one-million list until we successfully load the same 10K websites across all six browsers. This way, we populate 10K website visits in both rounds of web crawling.

Given that we use `mitmproxy` to capture HTTP traffic, we filter out any requests unrelated to a website visit, e.g., any browser-specific traffic that appears when it is opened. For example, Firefox downloads the latest blocking list when it is first loaded. To ascertain which requests are browser-specific, we find the intersection of web requests that persistently appear across 100 randomly selected website visits. We then manually validate these requests and mark them as browser-specific. Then, as a final step, we purge all such requests from our measurement studies to minimize the systematic noise in our data.

4 COMPARISON OF WEB TRAFFIC

In this section, we investigate the first research question, **RQ1: *What are the major differences in the web traffic generated for different mobile browsers?*** To answer this, we compare the traffic contents generated using different mobile browsers. This includes analyzing content types and their sources. We also analyze first and third-party content requests and cookies set. Lastly, we look at privacy-invasive web APIs accessed across different mobile browsers. Overall, this section provides a general comparison of privacy-intrusive web behavior across generic and privacy-focused browsers.

4.1 Differences in Web Requests

Content Types. The first difference in web traffic we observe is the type of content that is loaded into each mobile browser. We use the 'Content-Type' field in the response header to determine the requested content type. Figure 2 shows the distribution of contents served across all the browsers. We note that generally, *Images* were the most requested content, followed by *JavaScript* and *JSON* files, respectively. Note that one popular tracking technology includes injecting 1-by-1 pixels in websites [67]. As expected, we also see that generic browsers encounter higher counts of content requests than privacy-focused browsers in each content category generally. For example, Brave had 58% lower content requests than Chrome on average. This is consistent with the overall policy of mainstream browsers to restrict minimal content requests to maximize user experience. On the other hand, among privacy-focused mobile browsers, Focus had the highest number of content requests on average (26.5% and 7% higher than Brave and Duck Duck Go, respectively).

First-party vs. Third-party. To understand the first and third-party web interactions, we note the count of such requests across browsers and summarize the total count in Table 1. We note that Duck Duck Go had slightly higher first-party request counts. Of the 10K websites visited, Duck Duck Go had the highest first-party request count in 7,189 websites compared to other browsers. To ensure this behavior is not coincidental, we also compare the first-party requests encountered in our second round of data collection that took pla ce one month after the first round. In the second round of web crawl, Duck Duck Go again experienced the highest inflow of first-party requests in 7,110 websites. As a privacy-focused browser, Duck Duck Go was exhibiting anomalous behavior by allowing a higher influx of requests. Upon further investigation, we discovered that this had to do with how Duck Duck Go fetched some contents (e.g., favicon) on certain domains. In cases

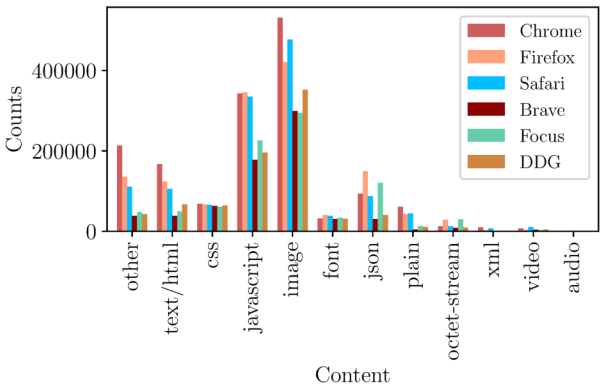


Figure 2: Difference in content-type distribution across different mobile browsers.

Table 1: Encountered request count and request type differences across different mobile browsers.

Browsers	First Party	Third Party
Chrome	428,439	**1,111,846**
Firefox	407,938	955,042
Safari	441,365	855,980
Brave	395,664	301,745
Focus	384,974	497,823
DDG	**466,187**	352,725

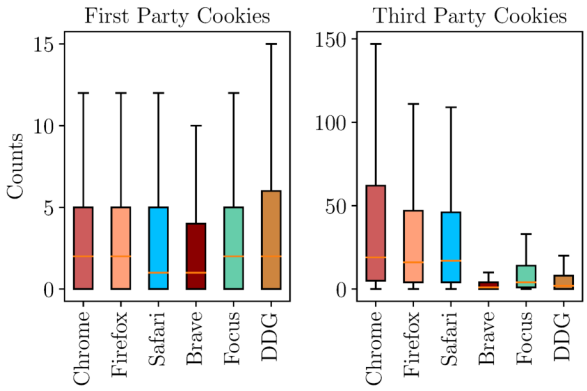


Figure 3: Box Plot for the distribution of first and third-party cookies that were set across different mobile browsers.

where a visiting domain does not provide its own `favicon.ico` and `apple-touch-icon.png`, Duck Duck Go will retrieve the file from its own server. However, this will enlist requests for such content under first-party bound requests. This abnormal behavior was later reported as a bug [22].

Cookies. Cookies are critical to user experiences and are extensively used by websites for bookkeeping and managing user interaction with the services it hosts. However, they are also used as a form of *stateful tracking*. We look at the number of first and third-party cookies that were set across each of the browsers. We then observe

Table 2: Top five third-party domains that set the highest number of cookies.

Brave	Duck Duck Go	Firefox-Focus	Chrome	Firefox	Safari
(facebook.net, 785)	(cookielaw.org, 621)	(googletagmanager.com, 3302)	(google-analytics.com, 5315)	(google-analytics.com, 5244)	(google-analytics.com, 5356)
(cookielaw.org, 624)	(youtube.com , 593)	(bing.com, 1043)	(doubleclick.net, 4429)	(doubleclick.net, 4327)	(doubleclick.net, 4400)
(youtube.com, 576))	(google.com , 506)	(cookielaw.org, 756)	(googletagmanager.com, 3515)	(googletagmanager.com, 3480)	(googletagmanager.com, 3520)
(adobedtm.com, 331)	(adobedtm.com , 354)	(youtube.com, 681)	(facebook.com , 3153)	(facebook.com, 3088)	(facebook.com, 3159)
(google.com, 320)	(bing.com, 293)	(tiktok.com , 371)	(adnxs.com, 2467)	(facebook.com, 2281)	(facebook.net, 2349)

the top services that set the highest number of third-party cookies in each browser.

We use two approaches to determine the cookies set in the crawl. First, we look for the 'Set-Cookie' field in the response header. Secondly, we use execution traces of JavaScript files and look for `document.cookie` function calls that had set the cookies. To distinguish between first and third-party cookies, we match the `eTLD+1` (effective top-level domain+1) of the request with the website domain visited. Figure 3 shows the distribution of the cookies. The distributions of first-party cookies reveal that Duck Duck Go had set the highest number of first-party cookies. This conclusion is consistent with the analysis of general first-party requests.

We also observe the top third-party domains that set cookies across each browser. This information is summarized in Table 2. We see a higher number of cookies set among generic browsers. Interestingly, all three generic browsers share the same top four domains that set the highest number of cookies. The top three of these domains (*google-analytics.com,doubleclick.net* and *googletagmanager.com*) are owned by Google. We also note that *googletagmanager.com* was allowed to set cookies in Focus in the same fashion as in the generic browsers. This is because while Focus blocks *google-analytics.com* by default, it only blocks *googletagmanager.com* when its privacy settings are set to strict mode.

Statistical Tests. We performed the Mann–Whitney-U test with a significance level of 0.05 (i.e., $\alpha = 0.05$). Our null hypothesis is: there is no significant difference between the distribution of requests among all pairs of browsers. We found that the $p-value < 0.05$, implying the null hypothesis was rejected. We performed the same test for content types and third-party requests and similarly observed significant differences in their distribution among mobile browsers. Such differences arise because each browser has its own mechanism for blocking content.

> **Finding 1**: *While generic browsers saw higher numbers of third-party content compared to privacy-focused browsers, for Duck Duck Go, a privacy-focused browser, we saw the highest number of first-party content (including cookies) across most of the websites crawled.*

4.2 API Usage

Browsers expose web APIs that are typically used with JavaScript to enable modern web applications. Among the myriad of such interfaces and web objects, certain APIs have been annotated as privacy-invasive across both desktop, and mobile platforms [38, 66]. In this section, we analyze to what extent privacy-invasive APIs are accessed by websites and scripts across different mobile browsers.

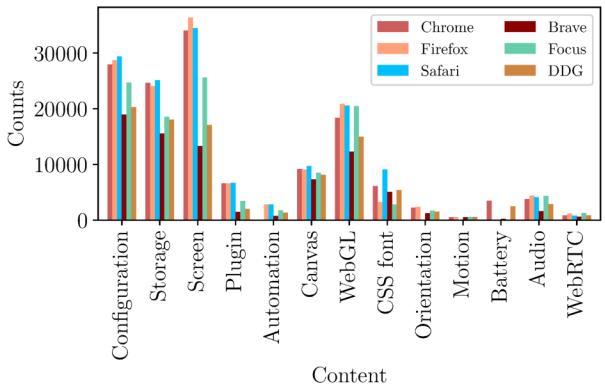


Figure 4: Counts of API usage across different mobile browsers.

The variations in API access can arise as different browsers may adopt different access control policies, e.g., blocking access at the request level for certain APIs [50].

To understand this better, we organize API calls into relevant categories. We adopt and modify the API to category mapping described by Cassel et al. [38], where we group the APIs into a higher-level functional context. We modify the mapping by updating the API stack, which forms different categories. For example, we add 17 more indicators for 'Automation' category that include new indicators for automation libraries like Selenium [32], Phantom [30], and nightmareJS [29]. Similarly, we add `BaseAudioContext` object under Audio. These small modifications ensure we properly represent each category from our dataset.

Figure 4 shows the resulting API access pattern. We observe that Focus encounters higher API accesses among all privacy-focused browsers across most categories. For instance, Configuration API accesses (that includes `navigator.userAgent`, `navigator.maxTouchPoints`, `navigator.languages`) is 30% higher in Focus than in Brave. These APIs provide useful information to websites but can also be used by fingerprinters to profile user devices by recording their device configurations and language preferences. One possible reason why these API accesses might be lower in Brave is that Brave adopts a very aggressive approach to avoiding fingerprinting by randomizing browser properties [24] or blocking such calls altogether, which is why many third-party trackers do not rely on such API calls.

We also find that Brave sees the lowest number of access to Storage APIs among the privacy-focused browsers (16.3% and 14% lower than Focus and Duck Duck Go, respectively). Brave claims to completely block all third-party application storage (e.g., cookies,

localStorage, indexedDB) and instead introduce its own partition storage mechanism called Ephemeral Third-party Site Storage [23].

Finding 2: *Due to aggressive third-party API blocking and fingerprint randomization, Brave sees the lowest number of API accesses across all categories.*

5 EFFECTIVENESS IN BLOCKING PRIVACY-INVASIVE CONTENTS

In this section, we investigate the following research question. **RQ2: *How effective are mobile browsers in blocking privacy-intrusive web content?*** From a user's perspective, a comparative analysis of such blocking behavior is important, especially as mobile devices have limited computing resources that can lead to existing countermeasures being optimized differently on mobile platforms compared to desktop platforms. To perform a comparative analysis, we compare how well our selected browsers emulate well-known anti-advertising and anti-tracking filter lists. If a browser's blocking mechanism closely emulates popular blocklists, it indicates a strong propensity towards privacy-protecting functionality.

In order to capture the effectiveness of the browsers, we choose well-known curated block lists like EasyList [11], EasyPrivacy [12], Disconnect [9], and WhoTracksMe [18]. These are popular anti-advertising and anti-tracking filter lists, and we will use them to assess the effectiveness of privacy-focused browsers. Being crowd-sourced and established in the privacy community, we considered these lists as the ground truth and defined them as baseline lists throughout the paper. The EasyList specifies the rules for blocking ads, whereas EasyPrivacy defines the rules for blocking trackers. Both lists perform prefix matching at the request level (i.e., URL). The tracker list for Disconnect and WhoTracksMe contains domain names with additional information like domain owner and domain category. Both of these lists perform blocking at the domain level. In order to capture the effectiveness of the browser under these baseline blocklists, we analyze if a request seen while using a given browser should have been blocked or not (we term such requests as *should-be-blocked*).

For the remainder of the paper, we classify each request as either *should-be-blocked* or *allowed* (i.e., not matching any filtering rule). To identify *should-be-blocked* or *allowed* requests, we have followed different techniques for different baseline lists. For EasyList and EasyPrivacy, we extended AdblockPlusFilterParser [1], which checks each request with a corresponding filter list through *regular expression* matching. From all the intercepted requests, we examine each request with the corresponding URL, referrer domain, content type, and third-party options to determine if the request should be considered as *should-be-blocked* or *allowed*. The same approach was used for Disconnect and WhoTracksMe. However, since both of these blocklists use domain-level blocking, we fetched the domain name from each request URL (eTLD+1) and matched it against the domain list provided by Disconnect and WhoTracksMe. If any matches were found, we consider it as *should-be-blocked*; otherwise, we label it as *allowed*. In addition, if the request URL and referrer contain the same domain, we considered it as *allowed* as it would not be a third-party request.

Table 3: Proportion of *should-be-blocked* requests across different mobile browsers.

Browsers	EasyList	EasyPrivacy
Chrome	14.7%	17.0%
Firefox	11.3%	14.9%
Safari	10.8%	14.8%
Brave	**0.13%**	**0.56%**
Focus	2.16%	7.86%
DDG	1.24%	4.21%

5.1 EasyList Filters

Easylist provides a suite of privacy filtration lists, each with its own functionality. The two most generic of these are EasyList and EasyPrivacy. We inspected all the observed requests to identify how effectively the browsers, especially the privacy-focused ones, have blocked requests from trackers or advertisers. To show contrast, we also show percentages of should-be-blocked requests from the generic browsers.

Table 3 summarizes the proportions of should-be-blocked requests. Generic browsers, especially Chrome, share the highest proportion of trackers and advertisers in our dataset. Safari was slightly better at blocking advertisers than Firefox and Chrome. Among all the browsers, Brave performs most closely to either of the lists as it uses these lists on top of its own heuristics to block requests [8]. However, despite using the list, it seems that Brave still allows some requests that should actually be blocked as per these lists. We investigate this further by analyzing the top third-party domains that Brave allows. These include *amazon-adsystem.com*, *doubleclick.net*, *googlesyndication.com* and *adnxs.com* and *facebook.net*. These domains belong to the Advertising and Analytics categories, according to Disconnect. It is possible that Brave's own heuristics makes some exceptions for these domains to uphold the website's functionality. For example, *facebook.net* issues a script called *fbevents.js* that is used to initialize and observe pixels to help websites gauge users' actions on their site. EasyPrivacy blocks this by default. However, Brave has made an exception for this rule for certain websites like *theonion.com*, *jacksonville.com* but not others.

Focus and Duck Duck Go, the other privacy-focused browsers, also generally performed well by blocking most of the should-be-blocked requests. In the case of Focus, the EasyPrivacy should-be-blocked percentage (7.86%) was slightly higher as it allowed script requests from services like *googletagmanager.com*. For both Focus and Duck Duck Go, *boost-next.co.jp* was one of the top services that should have been blocked (according to EasyList) but was not, whereas, in Brave, it was entirely blocked. *boost-next.co.jp* is an online marketing service that collects user information and stores persistent cookies [21]. Similarly, another tracker from Microsoft named *bat.bing.com/bat.js* is also classified as a tracker by EasyPrivacy. However, both Focus and Duck Duck Go had 3,983 and 1,131 (respectively) instances of this tracker not blocked. In comparison, Brave restricted the execution of this tracker to only 21 instances. Recently, Duck Duck Go offered an official explanation for allowing the presence of the tracker on its platform. Duck Duck Go stated that it could not block this tracker due to a policy requirement

Table 4: Categories of scripts identified by Disconnect. Bold numbers indicate lowest counts in that category across all browsers.

Browser	Email	General Fingerprinting	Disconnect	Content	Social	Analytics	Advertising	Invasive Fingerprinting	Email Aggressive
Chrome	119,208	215,002	157,087	143,961	359,649	54,282	14,305	9,098	3,151
Firefox	91,745	161,982	136,927	128,718	45,835	12,143	187,710	7,516	2,530
Safari	90,456	161,706	147,268	124,526	10,045	46,287	145,937	8,902	3,226
Brave	**5,273**	9,795	58,166	8,125	**683**	1,251	3,212	804	1,433
Focus	74,758	**2,732**	**9,341**	1,329	4,502	**661**	1,929	**594**	978
DDG	73,269	5,700	14,837	**757**	3,064	5,709	**1,524**	1,851	**964**

related to the use of Bing as a source for its private search results. However, Duck Duck Go later revoked the agreement and agreed to indiscriminately block trackers and advertisers from Microsoft, Facebook, Google, etc. [4].

5.2 Disconnect

We match all scripts recorded in our dataset against the services listed in Disconnect. In case of a hit, we assign the vendor of that script the category that is specified in Disconnect. Table 4 highlights these categories and the number of script requests that fell under those categories. We see that generic browsers were more accepting of vendors that perform fingerprinting, email-based tracking, analytics, and online advertising. Chrome had the highest exposure to such vendors as it does not block analytics and cross-site trackers by default. Generally, privacy-focused browsers take a more complex approach to blocking invasive content. For example, Brave utilizes its own heuristics, apart from using EasyList and EasyPrivacy, to block ads and trackers to limit the number of fingerprinting trackers [5]. Similarly, Duck Duck Go, which specializes in email protection [3], allowed the least number of Email and Advertising vendors. However, despite its claimed mission of not allowing itself or third-party vendors to collect user information, Duck Duck Go allowed the most number of Analytic trackers among all the privacy-focused browsers. We also note that Brave allowed a high number (9,795) of general fingerprinting script requests. It is important to note that Disconnect [9] distinguishes between general fingerprinting and invasive fingerprinting as the former only utilizes property and function accesses to collect information about users in unintended ways, whereas the latter category utilizes sophisticated fingerprinting techniques such as Audio and WebGL fingerprinting to profile user device in ways for which the APIs were not designed to be used. We experimentally confirm that the higher number of scripts in the Fingerprinting General category is attributed to first-party scripts, which generally perform important profiling of user devices for website customization and functionality adjustment. However, Focus recorded the lowest numbers in Invasive Fingerprinting and Analytics categories among all the privacy-focused browsers. This is because Focus uses the Disconnect list for privacy enhancement [59].

5.3 WhoTracksMe

Adguard maintains a tracking list that identifies trackers and advertisers and divides them into different categories, such as Advertising or Site Analytics. Figure 5 shows the distribution of script-based requests that were labeled using the WhoTracksMe list.

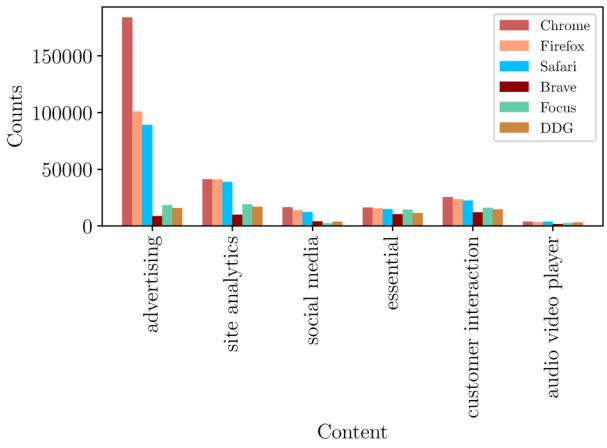


Figure 5: Counts of requests that were flagged by WhoTracksMe. Associated with these counts are the categories of vendors as defined by the list.

Among the generic browsers, we observe that Chrome had higher exposure to advertisers than Firefox and Safari. This is because of Firefox's Enhanced Privacy Protection [59] and Safari's Intelligent Tracking Prevention (ITP) [31] that is enabled by default. Both of these technologies claim to balance privacy and performance by blocking cross-site trackers and invasive advertisers while maintaining website functionality. Interestingly, Brave restricted the highest number of scripts associated with tracking. For example, it blocks 47% and 40% more Site Analytic trackers than Focus and Duck Duck Go, respectively. Similarly, Brave blocked 51.2% and 44% more Advertising scripts than Focus and Duck Duck Go, respectively. However, Focus appeared to block social trackers more aggressively than Brave. For instance, Brave allowed *facebook.net/fbevents.js* on 1,008 instances, whereas this tracker did not occur at all for both Focus and Duck Duck Go. Through these experiments, we discover the strengths and weaknesses of different privacy-focused browsers. Using blocking lists as ground truth, we were able to classify scripts into relevant categories and observed the efficacy of privacy-focused browsers in blocking or allowing privacy-infringing content.

Finding 3: *Brave outperforms Focus and Duck Duck Go by blocking most services and vendors classified as trackers by blocking lists, including top domains such as googletagmanager.com and bing.com. However, we observe that Focus blocks the highest number of social trackers, whereas Duck Duck Go leads in restricting email-based tracking.*

6 BROWSER-SPECIFIC SCRIPT ANALYSIS

Each browser supports its own suite of features and functions that can be used to interact with the website. Often at times, some functions may be available in one browser but not in another [2]. These APIs are critical for the browser vendor as they provide essential information that services can use to customize the user experience. For instance, declaring the number of logical cores through `navigator.hardwareConcurrency` can allow websites to optimize website functionality by displaying light-weight versions of the site for low-end devices [27]. However, opportunistic fingerprinters can also use these methods to profile users in invasive ways. Moreover, since each browser may expose a slightly different fingerprinting surface based on JavaScript engine or vendor-enforced configurations, trackers may use slightly different fingerprinting techniques for each browser.

In this section, we look to answer the following research question, **RQ3: *Do trackers discriminate against users based on the browser they use?*** To answer this question, we first detect invasive fingerprinters by utilizing the runtime APIs they access. We update and apply the heuristics provided by Englehardt et al. [43] (to account for changes made to the modern browser API stack) to detect the prevalence of fingerprinters across different privacy-focused browsers. Lastly, we apply static analysis on source codes and study the deviations in function calls and property accesses to answer whether trackers serve different versions of the same script based on what browsers a user uses.

6.1 Dynamic Analysis

WhoTracksMe [17] defines invasive fingerprinting as a technique that utilizes fingerprinting routines based on APIs to gather unique information about a device for which those APIs were not intended to be used. We use this definition to categorize opportunistic fingerprinters in our dataset. In order to find suitable heuristics that can be applied to identifying fingerprinters, we leverage the routines introduced by Englehardt [43]. These routines look for indicators of fingerprinters. For example, font fingerprinting is a popular tracking technique that creates a canvas object on the user's screen (most of the time, it is invisible, so the user is unaware) and then renders all possible fonts in its super list. When the user's device does not support a font, the default value is returned, which is how the fingerprinter knows that the font is unavailable. This font profiling results in a highly unique user trace as font selection is custom to each user.

Apart from font fingerprinting, other routines include WebRTC, Audio, and Canvas Fingerprinting. We use these routines to classify invasive third-party trackers. Table 5 summarizes the count of websites and each of the heuristics that matched. We note that Brave had the lowest number of websites where Audio, WebRTC, and Canvas fingerprinting occurred. This is due to the extensive third-party blocking of Brave and fingerprint randomization of API routines.

Table 5: Number of Tranco websites where potential trackers were found conducting the following fingerprinting routines. Bold numbers indicates the corresponding value is the lowest across all browsers. The "Overall" column records the total number of websites where at least one of the fingerprinting activity was observed.

Browsers	Audio	WebRTC	Canvas	Canvas Font	Overall
Chrome	344	117	960	**82**	1027
Firefox	398	114	1089	512	1527
Safari	379	84	1007	526	1442
Focus	382	115	1042	142	1147
Brave	**132**	**56**	**492**	98	**520**
DDG	244	100	776	98	827

Interestingly, Focus had a high number of websites where invasive fingerprinting occurred. For instance, on 1,042 websites visited by Focus, scripts were found to be using Canvas Fingerprinting. This performance was on par with Safari and Firefox, which also experienced a similar number of fingerprinters. To explore this further, we look at the scripts responsible for Canvas Fingerprinting in Focus. We notice that top vendors that perform Canvas in Focus belong to *salesforce.com* and *hcaptcha.com*. Both of these domains serve trackers that are blocked by the EasyPrivacy list.

Also, surprisingly, Chrome had the lowest number of websites where Canvas Font fingerprinting was observed. One possible reason for it could be that Canvas Font fingerprinting is a relatively expensive process where a script checks the availability of fonts by iteratively calling Canvas graphic operations. Since Chrome, as a generic browser, has an unrestricted fingerprinting surface, third-party vendors might be using other fingerprint profiling techniques that incur less cost. We also observe that roughly 10% and 15% of websites loaded in Chrome and Firefox, respectively, experience fingerprinting activity. Our reported numbers are on par with the previous works that have studied fingerprinting on the web [38, 50].

6.2 Static Analysis

Privacy-intrusive scripts may target users based on their browsers. The differences may arise due to a script's dynamic routines inside the browser. However, it may also arise because of differences in the script's content served to users of different browsers. In this section, we use static analysis to discover content-wise differences in the scripts loaded inside different privacy-focused browsers. We aim to discover how script authors may serve different codes based on browser type.

The proposed static analysis provides important information regarding the code structure. Specifically, it describes the function calls, variable declarations, and property accesses. We first aggregate scripts that were served across all three privacy-focused browsers to find variant behaviors among scripts. For fairness and ecological validity, we condition that a script may only be selected if it was seen in each browser when visiting the same website. For instance, while visiting a website X, a script Y was served in each privacy-focused browser; we call it a "common script". Moreover,

we match the complete script filename to avoid false positives (i.e., different scripts coming from the same vendor). An example of a script filename is: *https://cdn.pdst.fm/ping.min.js*.

There are 298,541 such common JavaScript programs in our dataset. It is noteworthy that the actual common script number may be higher, but it is hard to define an identifier (apart from the full script filename) that can be used to find commonality without introducing false positives. This is because script filenames may contain variable filenames where the last segment of the name (e.g., *ping.min.js*) may be similar. Therefore, we do not consider such scripts common.

To perform a differential analysis of the content of the served JavaScript programs, we consider common scripts with dissimilar content. Such scripts will henceforth be called "dissimilar" scripts. To find these, we convert the raw source code of common scripts into Abstract Syntax Tree (AST) representation. ASTs are tree-like structures, with each node representing a construction within the source code. Conversion to ASTs is important as it allows us to ignore whitespaces and variable name changes. Then, we compare the hashes of ASTs of common scripts and aggregate files with different hashes. This aggregation results in a set of dissimilar common scripts. In our dataset, we are able to find 5,729 such scripts in Brave, Focus, and Duck Duck Go.

Next, to identify content differences in these scripts, we leverage the list of important static code features from the work of Umar et al. [50]. These features are based on various code constructions like member expressions (e.g., navigator and math object call), literals, and variable declarations. We determine the presence of these important features in the common scripts. Intuitively, if the same script is being served across multiple platforms, it should have a similar feature presence. If not, this means that script vendors are deterministically serving changed scripts to different users.

Finally, we perform Agglomerative Clustering [19] on the dissimilar scripts based on the presence of these feature sets. Specifically, if a feature is present in a script, the indicator for that feature is turned on. Agglomerative Clustering is a bottom-up approach that starts with individual clusters and then determines the minimum distance required to merge with each other. All clusters meet at the root. For our experiment, we use 'Euclidean Distance' as the criteria for merging as it suits our simplistic input (feature presence). We note that the first cluster is formed between Brave and Duck Duck Go. The euclidean distance between Brave and Duck Duck Go is 63.3, whereas the euclidean distance between Brave or Duck Duck Go and Focus is 1,151. This suggests that Brave and Duck Duck Go users are likely to experience near-identical scripts. We also look at the count of feature indicators among the 5,729 common scripts and observe that Focus had 24 more `addEventListener` routines, 27 more `navigator` property access, and 82 more `sessionStorage` access points than Brave and Duck Duck Go.

Finding 4: *Script vendors may provide altered versions of JavaScript code to users based on their browser choice. Out of the privacy-focused browsers, Focus witnessed the most number of altered scripts. Duck Duck Go, and Brave had highly similar contents among the common script files.*

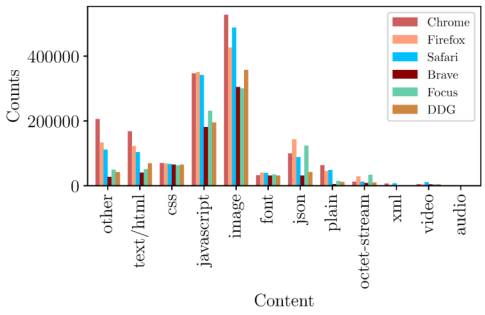
7 TEMPORAL VALIDATION

Contents served on websites often change. Such changes can arise as some vendors might update their scripts and content sources routinely. Moreover, the dynamic nature of content-loading means some domains may load a different set of scripts at different snapshots. To ensure that our conclusions and findings in this measurement study were not attributed to the timespan in which we gathered the data, we conducted a separate round of web crawl to validate the results. It should be noted that the first round of crawl took place in June 2022, and the second round of crawl took place in July 2022. This section aims to validate the consistency of our results in the earlier sections. We briefly replicate some of the key analyses discussed earlier and show that our conclusions are consistent regardless of the dynamics of web content. The figures and table we share in this section will hence be based on the data from the second round.

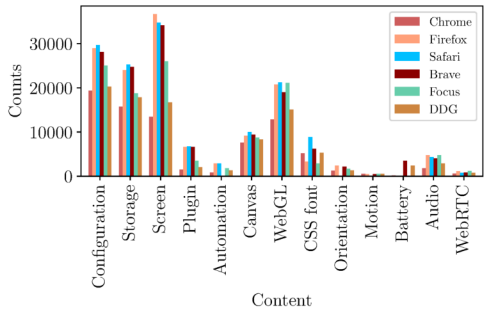
General Content Distribution. We start our validation by looking across the various content types loaded in the browsers in our two rounds of measurements. Figure 6a highlights the counts of content types that were loaded in the second round of crawl. We compare it with the counts of contents in the first round, as shown in Figure 2. We observe that there is no major change in the counts of these contents across the two crawls. To further validate this, we perform statistical testing on the content data from the two crawls. We select Mann-Whitney-U test [28] with a significance level of 0.05 (i.e., $\alpha = 0.05$). We specify our hypothesis as the following: there is no significant difference between the distribution of content types across the two crawls. We perform these tests on the samples of each browser across both crawls and find that the $p - value > \alpha$ (i.e., we can not reject the null hypothesis), meaning there was no statistically significant difference across the distribution of contents across the two crawls.

API Usage Pattern. While we record content and request/response headers using `MitmProxy` directly, the dynamic execution trace is recorded using a script-injection method. We recompute the API usage pattern from the second round dataset to ensure that the dynamic traces are consistent. Based on our earlier measurements, we concluded that in addition to blocking invasive fingerprinters, Brave restricts access to its fingerprinting surface to third parties. Figure 6b reinforces these conclusions as Brave again has the lowest counts across most categories of APIs. To validate that the distributions of values across crawls are not significantly different, we use Mann-Whittney-U test with a significance level of 0.05 (i.e., $\alpha = 0.05$). This test is suitable as it makes no underlying assumption about the distribution of the samples. We state our null hypothesis as the following: there is no significant difference between the distribution of API usage across the two crawls. We perform these tests on the samples of each browser across both crawls and find that the $p - value > \alpha$, meaning there was no statistically significant difference across the distribution of API usage across the two crawls. Figures 4 and 6b highlight similar distributions.

Should-be-blocked Requests. We use the EasyList and EasyPrivacy lists to compute the *should-be-blocked* requests. These lists are more comprehensive as they enforce request-level blocking. Table 6 highlights the percentages of should-be-blocked requests using the



(a) Content Type



(b) API Usage

Figure 6: Distribution of Content Type and API's in the second measurement round.

Table 6: Proportion of requests that should-be-blocked in round 2. The change over round 1 is shown in parenthesis.

Browsers	EasyList (Δ)	EasyPrivacy (Δ)
Chrome	14.2% (0.5% ↓)	17.18% (0.18% ↑)
Firefox	11.07% (0.23% ↓)	15.0% (0.1% ↑)
Safari	10.4% (0.4% ↓)	14.9% (0.1% ↑)
Brave	**0.04%** (0.09% ↓)	**0.44%** (0.12% ↓)
Focus	2.00% (0.16% ↓)	7.88% (0.02% ↑)
DDG	1.13% (0.11% ↓)	4.03% (0.18% ↓)

data from the second round. The numbers are comparable to the numbers found in the first round (shown in Table 3). We see that the maximum difference in change ranges from 0.02% to 0.23%. To statistically validate the distributions of should-be-blocked requests that appeared across the two rounds of crawls across the browsers, we again perform Mann-Whittney-U test. We select a significance level of 0.05 (i.e., $\alpha = 0.05$). We find that the $p - value > \alpha$ for both block lists, meaning there was no statistically significant difference across the distribution of should-be-blocked requests across the two crawls.

Finding 5: *Content distributions and dynamic runtime logs of APIs used by different browsers are consistent temporally. Our findings from earlier sections can be confirmed as each browsers web interaction remains unchanged across the two crawls.*

8 DISCUSSION

Browser vendors are rallying around a pro-privacy narrative and are introducing new technologies routinely to offer better privacy protection against advertisers and trackers. These technologies often balance protecting user information and maintaining a consistent and smooth web user experience. Among generic browsers, the goal is to maximize user experience while mitigating tracking exposures. For privacy-focused browsers, vendors may take a harder stand at thwarting tracking practices to protect user data. We know that vendors like Brave may even take unconventional approaches like API randomization or complete restriction. However, tracking companies have been known to use sophisticated tracking and user profiling techniques to thwart countermeasure techniques.

The objective of this study is to evaluate the privacy-protection capabilities of mobile devices. We use browsers from two major platforms (Android and iOS), including three privacy-focused browsers. We use real browsers to uphold ecological validity and use measurements from two separate rounds to ensure results are consistent. In this study, we look at the question of privacy protection from various vantage points and ask what general differences in web content are seen across browsers. We provide an overview of general differences in web traffic experienced by each browser using header files, content sources, and dynamic execution traces. We also ask how privacy browsers respond to third parties known to be trackers and advertisers by well-known block lists. These lists include EasyList, EasyPrivacy, Disconnect, and WhoTracksMe. We carefully choose these block lists based on their adoption among the privacy-conscious community that routinely maintains them. Thus, our analysis uses reliable community-sourced data to evaluate the effective performance of each browser. We note that Chrome experiences the most content among generic browsers that may harm user privacy in general. This is because, at default settings, Chrome offers minimal protection against cross-site tracking and third-party cookie tracking. The other two mainstream browsers, Firefox and Safari, have recently implemented some basic privacy enhancement technologies, such as Firefox's Enhanced Tracking Protection [59] and Safari's Intelligent Tracking Prevention (ITP) [31]. On the other hand, privacy-focused browsers implement more complex countermeasures and block any content classified as trackers. Each browser's approaches to information protection are different and result in unique strengths and weaknesses for the browsers. For example, Brave blocks many third-party domains classified as advertisers by EasyList and trackers by EasyPrivacy. Duck Duck Go performs especially well against third-party email trackers. Focus, on the other hand, blocks the highest number of requests from social trackers. It should be noted that while these privacy-focused mobile browsers implement these blocklists at various levels of privacy protection, these may not be wholly applied, as some browser vendors may have data-sharing agreements with some companies. We saw this with Duck Duck Go shared data with Bing [4].

Lastly, we define invasive fingerprinters and use state-of-the-art approaches to identify them in our dataset. We profile the behavior of such opportunistic fingerprinters using dynamic and static analysis of runtime execution logs and source codes, respectively. We ask whether certain browsers are more susceptible to tracking by such

scripts. We discover that Brave and Duck Duck Go generally limit their fingerprinting surface, thereby reducing information leaks to these fingerprinters. We confirm this by statically analyzing the functions and property accesses of common scripts whose content is dissimilar among privacy browsers. We note that Focus served the most altered script content as compared to Brave and Duck Duck Go. We also show evidence of tailored scripts and claim that third-party vendors may serve altered scripts to users based on browser choice. To ensure that our data samples are not a function of the timespan in which the measurement study is conducted, we perform another round of web measurement on these browsers and obtain similar distributions of content and API uses temporally.

Recommendations. We also provide some recommendations for privacy-focused browser vendors and their users to ensure no gaps are left to protect users' private data. We recommend that browser vendors leverage multiple blocklists to block ads and trackers that are commonly known. Vendors can also periodically send privacy checkup nudges to keep users informed about the latest privacy-enhancing features in the browser settings. Vendors should also be completely transparent about data-sharing relationships with other entities. For instance, Firefox Focus stores user data and shares it with the user's default search provider [25]. Users should be informed about such partnerships when they install the application. Lastly, many browsers, including generic ones, have implemented privacy features that are not enabled by default (for instance, Chrome allows blocking third-party cookies, but it is not enabled by default). These features should be enabled by default or, at the very least, be shown as an option to the user when they install the browser on their phones for the first time.

Limitations. We discuss some of the limitations of our measurement study. First, since the overarching goal of this paper is to study the interaction of browsers with trackers and advertisers, we ensure we have both static code source and runtime dynamic execution trace. However, to populate feature sets that complement static and dynamic analysis, we have to know beforehand which features we will look for in both the source code and the dynamic trace. As a result, we prepare an excessively rich feature list; however, it is by no means exhaustive and is subject to continuous improvement. Second, we perform dissimilarity analysis on invasive scripts by finding the common scripts that appeared in each privacy browser. While the script filename matching ensures we have no false positives, it can also result in false negatives. This results in a lower-bound number of common scripts for which we believe the actual number may be higher and can result in more interesting insights about deviant content among invasive fingerprinters. Third, we acknowledge that third parties can also set first-party cookies for web tracking, as demonstrated by [39]. We would need to instrument each browser and perform taint analysis on data types to track such events to determine if third parties were setting first-party cookies. However, since all of our tested mobile browsers are proprietary, such instrumentation is practically impossible. Furthermore, the smartphones used in this study were connected to the same lab network. Thus, from an external party, they might all see the same IP address making the requests. This can potentially lead to serving content based on tracking IP addresses. However, we crawled the same web page from smartphones in parallel to reduce the impact of generating drastically different content. Lastly, we do not consider inline JavaScript code (scripts embedded in a script tag within the main document page of a website) for dissimilarity analysis. Part of the reason for it is that while privacy browsers act hard on third-party trackers, they have minimal claimed protection against first-party content. However, analyzing the inline scripts for deviant content may also reveal interesting insights into first-party discrimination of users based on browsers.

9 CONCLUSION

In this paper, we present, to our knowledge, the first empirical measurement study to understand varying privacy protections offered by mainstream and privacy-focused mobile browsers. These browsers belong to the iOS and Android platforms and are selected based on their adoption rates. We compare the privacy-preserving behaviors of these browsers in terms of requested contents, used functions, and APIs. Further, we evaluate how closely privacy-focused browsers emulate known blocklists in blocking known trackers and ads. Through dynamic and static analysis, we also discover that different versions of scripts may be served deterministically based on the type of mobile browser. These selections may have privacy implications for users as some of the scripts contained invasive fingerprinting routines.

ACKNOWLEDGMENTS

We thank our anonymous reviewers for their valuable feedback. This material is based upon work supported in parts by the National Science Foundation under grant number CNS-2138138. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] 2016. *Adblock Parser.* https://github.com/bbondy/abp-filter-parser
[2] 2018. Browser Feature Comparison. https://caniuse.com/ciu/comparison.
[3] The Verge 2018. *Duck Duck Go Email Protection features.* The Verge. https://www.theverge.com/2021/7/20/22576352/duckduckgo-email-protection-privacy-trackers-apple-alternative.
[4] Duck Duck Go 2018. *More Privacy and Transparency for DuckDuckGo Web Tracking Protections.* Duck Duck Go. https://spreadprivacy.com/more-privacy-and-transparency/.
[5] Brave 2019. *Supporting The Web Privacy Community.* Brave. https://brave.com/supporting-the-web-privacy-community/
[6] Google Developers 2021. *Android Debug Bridge (adb).* Google Developers. https://developer.android.com/studio/command-line/adb
[7] 2021. *Blink.* https://www.chromium.org/blink
[8] 2021. *Brave Browser.* https://brave.com/
[9] 2021. *Disconnect.* https://github.com/disconnectme/disconnect-tracking-protection
[10] 2021. *DuckDuckgo: Privacy, simplified.* https://duckduckgo.com/app
[11] 2021. *EasyList.* https://easylist.to
[12] 2021. *EasyPrivacy.* https://easylist.to
[13] 2021. *Firefox Focus.* https://support.mozilla.org/en-US/kb/focus
[14] 2021. *Gecko.* https://developer.mozilla.org/en-US/docs/Mozilla/Gecko
[15] 2021. *OpenWPM-Mobile.* https://github.com/sensor-js/OpenWPM-mobile
[16] 2021. *mitmproxy is a free and open source interactive HTTPS proxy.* https://mitmproxy.org/
[17] Cliqz 2021. *Top most prevalent trackers on the web.* Cliqz. https://whotracks.me/trackers.html
[18] 2021. *WhoTracksMe.* https://whotracks.me/trackers.html
[19] 2022. Agglomerative Clustering. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html.
[20] 2022. Appium. https://appium.io/.

[21] 2022. Boost Next, Confection. https://confection.io/scripts/boost-next-co-jp/.

[22] 2022. [Bug] favicon.ico privacy leaks. https://github.com/duckduckgo/Android/issues/2004.

[23] Brave 2022. *Ephemeral Third-party Site Storage*. Brave. https://brave.com/privacy-updates/7-ephemeral-storage/.

[24] Brave 2022. *Fingerprint Randomization*. Brave. https://brave.com/privacy-updates/3-fingerprint-randomization/.

[25] Mozilla 2022. *Firefox Focus and Klar Privacy Notice*. Mozilla. https://www.mozilla.org/en-US/privacy/firefox-focus/.

[26] 2022. Google Play Ranking: The Top Free Tools Apps in the United States. https://www.appbrain.com/stats/google-play-rankings/top_free/tools/us/.

[27] Google 2022. *Google Web Light*. Google. https://developers.google.com/search/docs/advanced/mobile/web-light.

[28] 2022. Mann Whitney U Test (Wilcoxon Rank Sum Test). https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_nonparametric/bs704_nonparametric4.html.

[29] 2022. Nightmare: A high-level browser automation library. https://github.com/segmentio/nightmare.

[30] 2022. Phantom JS: Scriptable Headless Browser. https://phantomjs.org/.

[31] Apple 2022. *Safari's Intelligent Tracking Prevention*. Apple. https://www.apple.com/safari/docs/Safari_White_Paper_Nov_2019.pdf.

[32] 2022. Selenium. https://www.seleniumhq.org/.

[33] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. 2014. The Web Never Forgets: Persistent Tracking Mechanisms in the Wild. In *Proceedings of the 21st ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 674–689.

[34] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. 2013. FPDetective: Dusting the Web for Fingerprinters. In *Proceedings of the 20th ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 1129–1140.

[35] Mika Ayenson, Dietrich Wambach, Ashkan Soltani, Nathaniel Good, and Chris Hoofnagle. 2011. Flash cookies and privacy II: now with HTML5 and ETag respawning. *SSRN Electronic Journal* (07 2011). https://doi.org/10.2139/ssrn.1898390

[36] Tas Bindi. 2016. *Mobile and tablet internet usage surpasses desktop for first time: StatCounter*. ZDNet. https://www.zdnet.com/article/mobile-and-tablet-internet-usage-exceeds-desktop-for-first-time-statcounter/

[37] T. Bujlow, V. Carela-Español, J. Solé-Pareta, and P. Barlet-Ros. 2017. A Survey on Web Tracking: Mechanisms, Implications, and Defenses. *Proc. IEEE* 105, 8 (2017), 1476–1510.

[38] Darion Cassel, Su-Chin Lin, Alessio Buraggina, William Wang, Andrew Zhang, Lujo Bauer, Hsu-Chun Hsiao, Limin Jia, and Timothy Libert. 2022. OmniCrawl: Comprehensive Measurement of Web Tracking With Real Desktop and Mobile Browsers. *In Proceedings on Privacy Enhancing Technologies* 2022 (01 2022), 227–252. https://doi.org/10.2478/popets-2022-0012

[39] Quan Chen, Panagiotis Ilia, Michalis Polychronakis, and Alexandros Kapravelos. 2021. Cookie Swap Party: Abusing First-Party Cookies for Web Tracking. In *Proceedings of the Web Conference (WWW)*. 2117–2129.

[40] Anupam Das, Gunes Acar, Nikita Borisov, and Amogh Pradeep. 2018. The Web's Sixth Sense: A Study of Scripts Accessing Smartphone Sensors. In *Proceedings of the 25th ACM SIGSAC Conference on Computer and Communication Security (CCS)*. 1515–1532.

[41] Nurullah Demir, Matteo Große-Kampmann, Tobias Urban, Christian Wressnegger, Thorsten Holz, and Norbert Pohlmann. 2022. Reproducibility and replicability of web measurement studies. In *Proceedings of the ACM Web Conference (WWW)*. 533–544.

[42] Peter Eckersley. 2010. How Unique is Your Web Browser?. In *Proceedings of the 10th Privacy Enhancing Technologies Symposium (PETS)*. 1–18.

[43] Steven Englehardt and Arvind Narayanan. 2016. Online Tracking: A 1-million-site Measurement and Analysis. In *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 1388–1401.

[44] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W. Felten. 2015. Cookies That Give You Away: The Surveillance Implications of Web Tracking. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*. 289–299.

[45] Christian Eubank, Marcela Melara, Diego Perez-Botero, and A. Narayanan. 2013. Shining the Floodlights on Mobile Web Tracking — A Privacy Survey. https://masomel.github.io/static/pubs/s2p2.pdf

[46] Samuel Gibbs. 2016. Mobile web browsing overtakes desktop for the first time. https://www.theguardian.com/technology/2016/nov/02/mobile-web-browsing-desktop-smartphones-tablets.

[47] Alejandro Gómez-Boix, Pierre Laperdrix, and Benoit Baudry. 2018. Hiding in the Crowd: An Analysis of the Effectiveness of Browser Fingerprinting at Large Scale. In *Proceedings of the 27th International Conference on World Wide Web (WWW)*. 309–318.

[48] Michael C. Grace, Wu Zhou, Xuxian Jiang, and Ahmad-Reza Sadeghi. 2012. Unsafe Exposure Analysis of Mobile In-App Advertisements. In *Proceedings of the 5th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*. 101–112.

[49] Thomas Hupperich, Davide Maiorca, Marc Kührer, Thorsten Holz, and Giorgio Giacinto. 2015. On the Robustness of Mobile Device Fingerprinting: Can Mobile Users Escape Modern Web-Tracking Mechanisms?. In *Proceedings of the 31st Annual Computer Security Applications Conference (ACSAC)*. 191–200.

[50] Umar Iqbal, Steven Englehardt, and Zubair Shafiq. 2021. Fingerprinting the Fingerprinters: Learning to Detect Browser Fingerprinting Behaviors. In *Proceedings of the 42nd IEEE Symposium on Security & Privacy (S&P)*. 1143–1161.

[51] B. Kondracki, A. Aliyeva, M. Egele, J. Polakis, and N. Nikiforakis. 2020. Meddling Middlemen: Empirical Analysis of the Risks of Data-Saving Mobile Browsers. In *Proceedings of the 41st IEEE Symposium on Security and Privacy (SP)*. 810–824.

[52] Balachander Krishnamurthy and Craig Wills. 2009. Privacy diffusion on the web: A longitudinal perspective. In *Proceedings of the 18th International Conference on World Wide Web (WWW)*. 541–550.

[53] P. Laperdrix, W. Rudametkin, and B. Baudry. 2016. Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints. In *Proceedings of the 36th IEEE Symposium on Security and Privacy (S&P)*. 878–894.

[54] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS)*.

[55] Adam Lerner, Anna Kornfeld Simpson, Tadayoshi Kohno, and Franziska Roesner. 2016. Internet Jones and the Raiders of the Lost Trackers: An Archaeological Study of Web Tracking from 1996 to 2016. In *Proceedings of the 25th USENIX Security Symposium (USENIX)*.

[56] Tai-Ching Li, Huy Hang, Michalis Faloutsos, and Petros Efstathopoulos. 2015. TrackAdvisor: Taking Back Browsing Privacy from Third-Party Trackers. In *Proceedings of the 16th International Conference on Passive and Active Measurement (PAM)*. 277–289.

[57] Jonathan Mayer. 2011. FourthParty: Web Measurement Platform. http://fourthparty.info/.

[58] J. R. Mayer and J. C. Mitchell. 2012. Third-Party Web Tracking: Policy and Technology. In *Proceedings of the 33rd IEEE Symposium on Security and Privacy (S&P)*. 413–427.

[59] Nick Nguyen. 2018. *Latest Firefox Rolls Out Enhanced Tracking Protection*. https://blog.mozilla.org/blog/2018/10/23/latest-firefox-rolls-out-enhanced-tracking-protection/

[60] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. 2013. Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. In *Proceedings of the 34th IEEE Symposium on Security and Privacy (S&P)*. 541–555.

[61] Elias P. Papadopoulos, Michalis Diamantaris, Panagiotis Papadopoulos, Thanasis Petsas, Sotiris Ioannidis, and Evangelos P. Markatos. 2017. The Long-Standing Privacy Debate: Mobile Websites vs Mobile Apps. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*. 153–162.

[62] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, Christian Kreibich, and Phillipa Gill. 2018. Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. In *Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS)*.

[63] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. 2012. Detecting and Defending Against Third-Party Tracking on the Web. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 155–168.

[64] Suranga Seneviratne, Harini Kolamunna, and Aruna Seneviratne. 2015. A Measurement Study of Tracking in Paid Mobile Applications. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec)*.

[65] Vincent Toubiana, Arvind Narayanan, Dan Boneh, Helen F. Nissenbaum, and Solon Barocas. 2010. Adnostic: Privacy Preserving Targeted Advertising. *Proceedings of the 17th Network and Distributed System Symposium (NDSS)*.

[66] Zhiju Yang and Chuan Yue. 2020. A Comparative Measurement Study of Web Tracking on Mobile and Desktop Environments. *Proceedings on Privacy Enhancing Technologies* 2020, 2 (2020).

[67] Ahsan Zafar, Aafaq Sabir, Dilawer Ahmed, and Anupam Das. 2021. Understanding the Privacy Implications of Adblock Plus's Acceptable Ads. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (ASIA CCS)*. 644–657.