sciendo

Proceedings on Privacy Enhancing Technologies ; 2022 (1):227–252

Darion Cassel*, Su-Chin Lin, Alessio Buraggina, William Wang, Andrew Zhang, Lujo Bauer, Hsu-Chun Hsiao, Limin Jia, and Timothy Libert

OmniCrawl: Comprehensive Measurement of Web Tracking With Real Desktop and Mobile Browsers

Abstract: Over half of all visits to websites now take place in a mobile browser, yet the majority of web privacy studies take the vantage point of desktop browsers, use emulated mobile browsers, or focus on just a single mobile browser instead. In this paper, we present a comprehensive web-tracking measurement study on mobile browsers and privacy-focused mobile browsers. Our study leverages a new web measurement infrastructure, OmniCrawl, which we develop to drive browsers on desktop computers and smartphones located on two continents. We capture web tracking measurements using 42 different non-emulated browsers simultaneously. We find that the third-party advertising and tracking ecosystem of mobile browsers is more similar to that of desktop browsers than previous findings suggested. We study *privacy-focused* browsers and find their protections differ significantly and in general are less for lower-ranked sites. Our findings also show that common methodological choices made by web measurement studies, such as the use of emulated mobile browsers and Selenium, can lead to website behavior that deviates from what actual users experience.

Keywords: Web tracking, web measurement, user privacy, mobile browsers, tracking protection

DOI 10.2478/popets-2022-0012
Received 2021-05-31; revised 2021-09-15; accepted 2021-09-16.

***Corresponding Author: Darion Cassel:** Carnegie Mellon University, E-mail: darioncassel@cmu.edu
Su-Chin Lin: National Taiwan University, E-mail: r07922067@ntu.edu.tw
Alessio Buraggina: University of Miami, E-mail: alessioburaggina@gmail.com
William Wang: University of Chicago, E-mail: williamwang@uchicago.edu
Andrew Zhang: University of Illinos Urbana-Champaign, E-mail: alz2@illinois.edu
Lujo Bauer: Carnegie Mellon University, E-mail: lbauer@cmu.edu
Hsu-Chun Hsiao: National Taiwan University, E-mail: hchsiao@csie.ntu.edu.tw

1 Introduction

When a user browses a website, their behavior is typically recorded by both the website itself and third-party scripts. The web tracking technology used by these sites has evolved from using cookies to using stateless methods (i.e., fingerprinting), where characteristics of the browser or the hardware it runs on are measured by scripts running in web pages to identify the browser out of millions of other browsers [8, 26, 42, 56, 57, 83, 102]. Data gleaned from tracking is most often used for targeted advertising but also has other uses, such as fraud prevention. Regardless of motive, web tracking is widely recognized as a persistent threat to online privacy as web browsing habits often reveal sensitive personal information [87]. Multiple studies have measured the extent of web tracking [19, 33, 43, 61]. Their results show a landscape of pervasive tracking, motivating the need to develop tools to mitigate tracking, such as ad and tracker block lists [12, 36, 41], ad-blocking extensions [18, 29, 46], and *privacy-focused* browsers [23, 30, 37, 76, 98].

Mobile browsers have become the dominant web browsing platform, overtaking desktop browsing in market share as of 2016 [92]. Mobile browsers present additional avenues to track users due to their JavaScript-accessible sensor APIs, such as those for orientation and motion. These sensor APIs give scripts access to precise hardware-dependent information about a device that could be used for fingerprinting. For instance, Das et al. showed that motion sensors could be used to generate unique fingerprints [34]. Further, the content served to mobile browsers can differ greatly from that served to desktop browsers. WTPatrol found that 25.9% of sites in a sample of the Alexa top 1 million had mobile-specific pages with large structural devia-

Limin Jia: Carnegie Mellon University, E-mail: liminjia@cmu.edu
Timothy Libert: Google, E-mail: timlibert@cmu.edu

tion from their desktop counterparts [106]. Studying web tracking in the context of mobile browsers is an important topic and is the subject of several recent projects [14, 33, 45, 53, 106].

Existing measurement studies either use emulated mobile browsers [33], which can differ significantly in behavior from real mobile browsers, or focus on mobile Firefox [45, 106], which is used by far fewer people than Chrome. Hence, there is still a need for a comprehensive study of web tracking on mobile browsers that: (1) analyzes data from a variety of browsers, including browsers with the largest market share and privacy-focused browsers; and (2) uses a crawling infrastructure that makes it more difficult for tracking scripts to recognize that crawled pages are visited by automated infrastructure and not by humans (e.g., tracking scripts are allowed to read real phone sensor data).

Testing with both mainstream and privacy-preserving browsers is important for several reasons. Browsers differ in what APIs they support and how that support is implemented (e.g., Chrome and Firefox can return different results for the `DeviceMotion` API [70]), potentially causing different tracking behavior. Hence, it is important for tracking measurement results to include browsers that are most commonly used. Consequently, our measurement includes Chrome, which has nearly 135 times greater market share than mobile Firefox and is employed by 63% of users [94]. Further, in a desktop setting it is common to evaluate the impact of tracker- and ad-blocking extensions and browsers [85]. Similarly, some mobile browsers claim to enhance user privacy, including by limiting tracking, but their relative strengths and weaknesses have not yet been carefully measured. We study several such browsers that have each been downloaded more than a million times.

As important as the selection of browsers is the methodology and infrastructure for collecting data (feature 2). Prior work has largely used Selenium [15] to operate browsers; however, without careful modification Selenium has been shown to be detectable by websites [54]. Another platform used by previous work, OpenWPM-Mobile [33], uses emulated mobile browsers in lieu of running browsers on real mobile phones. Mobile browsers are emulated by loading pages in a desktop browser and spoofing the browser's responses to JavaScript API calls to look like those of a mobile browser. This has practical advantages, but since emulated mobile browsers are not backed by real phones, their behavior can diverge from the hardware-dependent fingerprints of real phones [20, 88]. For example, the Canvas fingerprint, which is GPU-dependent, may be different. These methodological choices could affect the ecological validity of results.

Just as the data-collection methodology can affect the veracity of any results, so can the method by which collected data is analyzed and synthesized into high-level findings. For example, data may be noisy enough that computing and comparing aggregate statistics such as averages may not be a reliable method for comparing two distributions. To increase confidence that any findings are solidly supported by data, our analyses rely on statistical tests that are common in other domains (e.g., clinical medical studies) but less often used in web measurement.

With a robust statistical testing pipeline, our work aims to investigate several questions. First, we evaluate infrastructure and methodological design choices: **Q1a**: Is it ecologically valid to use emulated browsers in a web measurement study? **Q1b**: Can the method by which the browsers are driven impact results? **Q1c**: Can content be equalized between desktop and mobile browsers? Second, we conduct a comprehensive comparison of web tracking between mobile, desktop, and *privacy-focused* browsers: **Q2a**: How do mainstream mobile and desktop browsers compare in terms of tracking and advertising? **Q2b**: How effective are privacy-focused browsers at blocking tracking and advertising? **Q2c**: Do privacy-focused browsers differ in effectiveness on mobile and desktop? **Q2d**: What are the strengths and weaknesses of individual privacy-focused browsers? **Q2e**: How does location affect tracking behavior?

To answer these questions we developed OmniCrawl: a comprehensive, cross-platform, web-privacy measurement infrastructure that leverages desktop computers and smartphones located. Using OmniCrawl, we perform a web crawl using seven mobile browsers: Chrome and Firefox, as well as five popular privacy-focused browsers: Brave, Tor, Firefox Focus, DuckDuckGo, and Ghostery. In addition to comparing individual browsers, we also compare against the desktop counterparts of four of these mobile browsers: Chrome and Firefox, which collectively hold 70% of browser market share [82], and the privacy-focused Brave and Tor. We visit 20,000 websites with a total of 42 browser instances running in parallel across four desktop computers and 18 mobile phones on two continents.

We find that using an emulated mobile browser, as well as some ways of using the popular Selenium browser driver, can each cause statistically significant differences in observed tracking behavior. We also find that while mobile browsers exhibit fewer tracking and advertising requests than desktop browsers, the same entities are

responsible for tracking and advertising on mobile and desktop browsers. Privacy-focused browsers are effective at reducing tracking and advertising requests, but their performance varies substantially. We also demonstrate a new heuristic for detecting ad-blocker evasion via registration of seemingly unrelated domains.

In summary:

- We develop and make publicly available (at https://github.com/OmniCrawl) a scalable web-crawling infrastructure capable of simultaneously visiting websites with multiple non-emulated desktop and mobile browsers. This is an improvement over prior tools that supported just a single real mobile device.
- We evaluate common methodological choices for measuring tracking on mobile web browsers and find that some choices can negatively impact the ecological validity of a study.
- We compare the amount of tracking experienced by users of popular desktop web browsers with the amount of tracking experienced on the corresponding mobile browsers and find that the third-party ecosystems of both platforms are more homogenous than previously approximated.
- We evaluate the efficacy of privacy-focused browsers' tracking protections and find them less effective against less common tracking entities than against more common ones. We also find significant differences in the behavior of individual privacy-focused browsers.
- We present a heuristic for detecting ad-blocker evasion by registering seemingly unrelated domains.

2 OmniCrawl Infrastructure

To enable our web tracking measurement to run robustly on real phones across two different continents, and synchronously collect data from multiple browser instances, our crawling infrastructure needed to surmount a number of engineering challenges: (1) Provisioning and managing multiple physical Android phones in an automated way; (2) Synchronizing crawling between browsers; (3) Ensuring the infrastructure does not use components that can negatively impact ecological validity; (4) Aggregating and preprocessing the large amount of data generated by each browser.

We considered several tools for measuring web tracking (Section 5.3). Most of these tools do not by default support simultaneous crawling with multiple real mobile and desktop browsers and thus do not solve the above engineering challenges. OpenWPM-Mobile [33] comes close to supporting parallel crawling with multiple browsers, but has one key limitation: it uses emulated mobile browsers rather than mobile browsers running on real mobile devices. As we show in Section 4.1.1, this design choice can lead to divergence in website behavior from what a real mobile browser would experience. As a result, we decided to create our own tool.

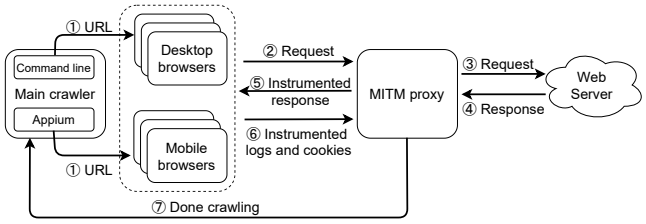

Fig. 1. OmniCrawl overview and workflow. All browsers are running on physical computers or mobile phones.

We built OmniCrawl, which supports multiple physical devices and browser instances while allowing large-scale web measurement across geographically distributed locations. In our study, we ran 42 browsers simultaneously, on 18 Android phones and two desktops, in two locations. The design and workflow of OmniCrawl is illustrated in Figure 1. The central controller of OmniCrawl is the *main crawler*, which controls desktop browsers through custom bash scripts and mobile browsers through Appium. We use custom scripts instead of Selenium [15], a web automation tool widely used by previous work [8, 33, 43], because websites may be able to detect the use of Selenium via the existence of modified JavaScript properties [88]. Our evaluation confirms that tracking observed on Selenium-driven desktop browsers differs from tracking on non-Selenium-driven browsers (Section 4.1.2). For mobile browsers, we build our own automation tool using Appium and Android Debug Bridge (ADB). We treat a mobile browser as a generic Android application and use Appium's *Go to URL* function to navigate to websites. The main crawler also ensures that all browsers access the same website within 130 seconds, reducing the influence of dynamic web content on our study. Our evaluation shows no significant difference in website content due to variability of load time within this window (Section A.2).

To scalably support multiple browsers, OmniCrawl uses mitmproxy [32] to intercept traffic and inject scripts to instrument JavaScript APIs. We install a custom root certificate on the mobile phones and desktop computers so the proxy can decrypt HTTPS traffic. We store all the request and response payloads except media files (e.g., images, fonts, and videos) larger than 4 KB.

This allows us to keep useful data such as JavaScript files while remaining efficient and conserving disk space.

JavaScript instrumentation is injected to each loaded HTML page and iframe. Compared with the instrumentation used in previous work [33, 43, 66], OmniCrawl instruments a more comprehensive set of APIs (including Chromedriver automation, math, XMLHttpRequest, fetch, and WebGL APIs) and logs API function arguments and return values, which are helpful for identifying fingerprinting techniques. For example, we heuristically identify canvas and WebGL fingerprinting (Appendix A.1) based on the canvas size and the image data embedded in the function argument and return value. Table 2 lists the APIs we instrumented that are most pertinent to fingerprinting. Besides motion, we also instrumented mobile sensor APIs such as magnetometers, proximity, and ambient light in accordance with prior work. These APIs were once supported by some browsers but have been removed or disabled in all browser versions we tested due to privacy concerns [73–75]. A limitation of an instrumentation-based infrastructure [33, 43, 66] such as ours is that the instrumentation may be detectable by the page scripts' checks for certain properties, e.g., the variables and functions defined in global scope by the instrumentation. While this can affect ecological validity, we reduce the risk to validity by using custom driver scripts and real mobile browsers; an effort, to our knowledge, only matched by two studies [45, 106] that only support Firefox.

To allow execution of OmniCrawl's injected inline script on pages with strict Content Security Policies (CSPs), we intercept and automatically rewrite the CSP header. If a nonce or hash directive is present in the CSP header, we append the nonce or hash of our injected script; otherwise we append an *unsafe-inline* directive. While adding *unsafe-inline* may allow the execution of unwanted inline scripts that are already on the page and may change the page's behavior, the impact on our results is negligible: for the 20K websites we visited, we added *unsafe-inline* to the CSP headers of only three.

3 Methodology

In this section we discuss our choice of browsers and explain our crawl configuration, resulting dataset, and the statistical methodology we use in our analysis.

Category	*Browser*	*Notes*
Desktop Browsers	Chrome 88.0.4324.0	-
	Chrome 88.0.4324.0	Identical for comparison
	Chrome 88.0.4324.0	Scroll down after 45 seconds
	Chrome 88.0.4324.0	Driven by Selenium
	Firefox 86.1.1	-
	Firefox 86.1.1	Driven by Selenium
	Firefox 86.1.1	Ghostery extension v8.5.5
	Firefox 45.9.0	-
Privacy-focused	Brave 1.20.103	-
	Tor Browser 10.0.12	Not using Tor network
Emulated Mobile Browsers	Firefox 45.9.0	OpenWPM-mobile [33]
	Firefox 86.1.1	OpenWPM-mobile
Mobile Browsers	Chrome 88.0.4324.181	-
	Chrome 88.0.4324.181	Spoofed desktop user-agent
	Chrome 88.0.4324.181	Scroll down after 45 seconds
	Firefox 86.1.1	-
Privacy-focused	Brave 1.20.103	-
	Tor Browser 10.0.12	Not using Tor network
	Ghostery 1.0.2033	Build 22251829
	DuckDuckGo 5.77.2	-
	Firefox Focus 8.13.1	-

Table 1. The browsers used in our crawl. We use the latest version of each browser as of February, 2021.

3.1 Browser Selection

To examine differences between tracking on mobile and desktop (**Q2a**), we chose two mainstream browsers (Chrome and Firefox) for an accurate picture of the typical user's experience. To understand the effectiveness of browsers that claim to enhance privacy as their primary feature (**Q2b-d**), we chose several popular browsers from the Google Play Store that make this claim.

For desktop browsers, we chose Chrome, Firefox, and two privacy-focused browsers, Brave and Tor, which have 15M [22] and 2M [99] monthly users, respectively. We use two identical Chrome instances to measure the baseline variability between visits to the same website (Appendix A.2). We included Selenium-driven versions of Chrome and Firefox to study the differences between emulated and real browsers (**Q1b**; Section 4.1.2). We include one version of Chrome that scrolls down the page in the last 45 seconds of a site visit (**Q2c**; Section 4.1.3). We run one instance of Firefox with the Ghostery plugin [29] to compare with the mobile Ghostery browser, based on Firefox. Firefox 45 was included to compare with the emulated mobile browser OpenWPM-Mobile, based on Firefox 45 (**Q1a**; Section 4.1.1). We summa-

rize our browser selection and configuration in Table 1. All browsers use default settings (e.g., whether to allow third-party cookies) loaded upon installation. During installation, if prompts appeared (e.g., whether to share analytics data), defaults were accepted.

For mobile browsers, we use counterparts to the desktop versions of Chrome and Firefox, and five popular browsers that claim to enhance privacy as a primary feature on their Google Play Store pages: Brave [23], Tor [98], Firefox Focus [76], DuckDuckGo [37], and Ghostery [30], each with 1M+ to 10M+ downloads as of Feb. 2021.

3.2 Measurement Setup

We crawled 20,000 websites from two different locations between February and June, 2021.

Website List. We selected websites from the Tranco website ranking[1] [58]. The Tranco ranking is designed to be manipulation resistant and stable over time, and thus is more representative than the Alexa website ranking. We crawled the top 10,000 Tranco websites and 10,000 lower-ranking websites randomly selected among every 10 sites ranked from 10,000 to 110,000 to ensure our dataset contains representative lesser-known websites.

Seed Profiles. A browser *profile* is a set of cookies and local storage representing a user's client-side state. Like prior work [43], we built a *seed profile* for each browser and reset the browser state to it before each website visit so that order of website visits does not affect the data we collect. Following prior work [43], we visited the Tranco top 1K websites using every browser in parallel and saved the resulting profiles as seed profiles. We chose the top 1K sites after small-scale experiments: We crawled the Tranco top 5K on desktop and mobile Chrome and measured the number of unique third-party domains that appear on more than five websites. Visiting the top 1K sites accounted for 87% of those domains, after which there was little increase.

Website Loading Time. Upon visiting a site, we allowed 90 seconds for the browser to load its content. This time was determined through a small-scale experiment that found that for most websites, the number of new requests diminished significantly after the 90-second mark. After this 90-second soft timeout, we redirected each browser to the next site. If a browser was unresponsive, we allowed for a maximum of 130 seconds total to pass before restarting the browser. We synchronized the browsers (e.g., ensured that they were all set to visit the same site next) after every ten sites.

Locations. We collected data from two different locations: the United States, in North America (NA), and Taiwan, in Asia (AS). Both were rated "Free" in the 2018 Freedom on the Net report [47]. Both locations are in a university network. We set up nine Android 8.1 Motorola G5 Plus in NA, and nine Android 8.1 ASUS ZenFone Max (M2) in AS.[2] All desktop browsers were on a Windows 10 machine, as the majority of desktop users use Windows [82]. The emulated mobile browsers were on an Ubuntu 18.04 LTS machine, with screen sizes set to be identical to the Android phones.

Data Gathered. For each website that was visited during the crawl, our infrastructure intercepted and recorded all requests that were made and JavaScript browser APIs that were accessed, including the count of accesses, by the website itself and by iframed content. We are able to distinguish first-party and third-party requests, and we log which script performed an API access and separate the accesses into 13 categories (Table 2). Distinguishing API accesses in these categories allows us to understand precisely what type of information is being collected by page scripts.

3.3 Data Preprocessing and Classification

Prior to analysis, we removed requests that were initiated by browsers rather than websites and determined the type (e.g., first-party, third-party, tracking and advertising) and provenance of each request.

To find requests that are browser-generated, we count how many web sites we observe each third-party URL. We then manually examine all third-party URLs that were requested by at least 50 websites on just one browser but not others. Through this we identify requests that are browser-generated, such as requests that Chrome browsers send to Google services.

Requests are classified as either first-party or third-party based on the comparison of the domain of the request to the domain[3] the request originated on. We also classified requests as tracking-and-advertising by refer-

1 Available at https://tranco-list.eu/list/4ZWX.

2 Motorola G5 Plus was no longer available in Asia at the time we began our study.

3 *eTLD+1*; Public suffix with one additional label (e.g., *example.com*) [77].

API category	*Instrumented objects*
Canvas	HTMLCanvasElement
Screen	screen
Motion	DeviceMotionEvent, Accelerometer, Gyroscope, LinearAccelerationSensor
Battery	BatteryManager
Storage	localStorage, sessionStorage, indexedDB
Audio	AudioContext, OfflineAudioContext, OscillatorNode, AnalyserNode, GainNode, ScriptProcessorNode
WebRTC	RTCPeerConnection
Automation	navigator.webdriver, chrome.app, document.cdc_asdjflasutopfhvcZLmcfl_
WebGL	WebGLRenderingContext, WebGL*
Plugin	navigator.plugin
CSS font	{CSSStyleDeclaration,CSS2Properties}.font*
Orientation	screen.orientation, DeviceOrientationEvent, AbsoluteOrientationSensor, RelativeOrientationSensor
Configuration	navigator.{platform,userAgent,appCodeName, appName,appVersion,maxTouchPoints}

Table 2. Instrumented JavaScript fingerprinting API categories.

ring to community-maintained ad and tracker-blocking lists: EasyPrivacy [41] and AdGuard's Tracking Protection Filter [12] for tracking; a variant of EasyList (without filter rules for adult domains) [40], AdGuard's Base Filter [9], and AdGuard's Mobile ads filter [11] for advertising. We also include an Asia-specific block list from AdGuard in order to better identify tracking and advertising in Country B [10]. Like related work [43, 66], we do not have knowledge of the server-side behavior of websites; instead we rely on these block lists to approximate the extent of tracking and advertising.

Finally, for every request, we used *webXray* [62] data to determine the provenance of the requests (e.g., which company owns the domain associated with the request). For example, a request to *youtube.com* is considered to belong to Google, the parent company of YouTube. For requests whose domains did not have an entry in webXray's dataset, the owner is the domain itself.

3.4 Fingerprinting Heuristics

We define five fingerprinting categories, shown in Table 3. Those fingerprinting categories are used in a popular open-sourced fingerprinting library *fingerprintjs2* [101], as well as in prior fingerprinting studies [24, 42]. The fingerprinting behaviors in each category are detected using heuristics applied to a scripts' API access patterns. For example, to detect font fingerprinting we check for scripts enumerating system fonts. All five heuristics are at least as strict as those defined in related work; details of each are in Appendix A.1.

Category	*Description*
WebRTC	Extract private IP address [33, 43]
Audio	Extract info. from underlying audio stack [33, 43, 56]
Font	Enumerate system fonts [8, 43, 83]
Canvas	Render text and 2D objects on CanvasRenderingContext2D [7, 33, 43, 57, 69]
WebGL image	Render text and 2D/3D objects with GPU on WebGLRenderingContext [26, 69]

Table 3. JavaScript fingerprinting heuristics categories. Detailed heuristics are presented in Appendix A.1.

3.5 Statistical Methodology

For statistical analysis we used standard tests (commonly used in medical research (e.g., [39, 90]) as well as privacy studies (e.g., [16, 21, 59, 63, 80, 91])). The distributions we examined were sufficiently skewed such that non-parametric tests were applicable. Thus, for each kind of comparison (two groups, multiple browsers, etc.) we used the appropriate non-parametric statistical test. We used the Wilcoxon signed-rank [104] and Mann-Whitney U [64] tests for comparisons of two groups of equal and unequal size, respectively. When we performed multiple tests, we made a testing correction using the Holm-Bonferroni method [51]. We used the Friedman test for comparison of more than two groups (omnibus) [48]. When the Friedman test showed a significant difference between some of the groups, we applied the Conover Squared Ranks test to determine which groups were significantly different [31].

4 Results

In this section we present findings on the two sets of research questions: methodological design choices (**Q1a-c**), and comparisons of mobile, desktop, and privacy-focused browsers (**Q2a-e**). Except for location comparisons (**Q2e**), comparisons of individual browsers use the NA dataset to avoid introducing location as a variable.

4.1 Evaluation of Methodological Choices

The methodological design choices of a web measurement study can impact the ecological validity of results. We extend previous work by investigating several dimensions in parallel: **Q1a**: Is it ecologically valid to use emulated browsers in a web measurement study? **Q1b**: Can the method by which the browsers are driven impact results? **Q1c**: How can content be equalized between desktop and mobile browsers, and what effect does this have on web-privacy measurements?

4.1.1 Validity of Emulated Mobile Browsers

An emulated mobile browser is a desktop browser modified to appear to websites as a mobile browser. OpenWPM-Mobile used this kind of browser in lieu of a mobile browser running on a mobile phone [33].

Q1a: Is it ecologically valid to use emulated browsers in a web measurement study?

Prior work has shown that using a desktop browser configured to impersonate a mobile browser may affect tracking measurement [106]: many Alexa Top 100 pages exhibited differences in the DOM and JavaScript content shown in an emulated mobile browser compared to a real mobile browser. However, the emulation in that work was limited to changing desktop Firefox to use the User-Agent string and screen resolution of mobile Firefox. Other modifications were not implemented, e.g., sensor API results were not spoofed, although they are often used to fingerprint mobile devices [20].

In comparison, OpenWPM-Mobile modifies browser properties and spoofs sensor API calls to the point that the browser fingerprint is almost identical to real mobile Firefox, except for the Canvas and WebGL fingerprints, which are difficult to emulate as they depend on the underlying graphics hardware [33]. While Selenium-driven, OpenWPM-Mobile avoids some of the potentially problematic browser modifications that Selenium makes (Section 4.1.2). Such deep modifications require regular updating as browser and web APIs change.

We included both OpenWPM-Mobile's emulated Firefox browser and its non-emulated counterpart, mobile Firefox, in our crawl. Our findings indicate that these two browsers' distributions of first-party, third-party, and third-party tracking-and-advertising requests all differ significantly (Figure 12). The median number of first-party requests is the same, but OpenWPM-Mobile has significantly *more* third-party (3%) and third-party tracking-and-advertising (6%) requests on average. However, the entities themselves are similar (the top 5 are Google, Facebook, Adobe Systems, Microsoft, and Amazon for both browsers). This mirrors what we see when comparing mainstream mobile and desktop browsers (Section 4.2.1): different distributions of number of entities, but the entities themselves are largely the same.

We also measured the number of API accesses by the pages they browsers loaded. The browsers differ significantly in the numbers of APIs accessed for all API categories besides Audio and WebRTC (Figure 13). For some APIs implicated in fingerprinting, the differences are fairly large: Plugin (50% more on OpenWPM-Mobile), Storage (21%), and Screen (8%). The Plugin category has the largest difference because those APIs are primarily used on desktop; support for plugins on Firefox for Android was removed in 2016 [71]. Websites may be treating OpenWPM-Mobile as a desktop browser and thus trying to access Plugin APIs.

We further examined websites in the Tranco top 500 that showed the largest differences: *alipay.com, engadget.com, yandex.ru, fideity.com, 6.cn, avito.ru.* Investigating the JavaScript loaded by these sites uncovered multiple forms of fingerprinting, including Canvas fingerprinting. OpenWPM-Mobile cannot emulate the Canvas fingerprint of a mobile device because it is using the graphics hardware of a desktop machine [33]; these websites may be recognizing OpenWPM-Mobile as a desktop browser. These findings suggest that even a carefully emulated browser like OpenWPM-Mobile can cause significant differences in tracking measurements, potentially due to difficult-to-emulate hardware differences from its real counterpart.

> **Result 1a**: Emulated browsers may not be suitable for use in a web privacy studies that measure counts of requests and accesses to browser APIs because their use can result in measurements that differ from real mobile browsers.

4.1.2 Method of Driving Browsers

A crawling infrastructure typically interfaces with a browser through a *driver* program, which may modify the browser in a manner detectable to websites.

Q1b: Can the method by which the browsers are driven significantly impact results?

Selenium [15] is a popular method for controlling browsers. However, previous work has shown that Se-

lenium is detectable by websites since it injects and modifies JavaScript properties of the browser [54, 88]. Hence, we use custom scripts to launch and direct browsers without modifying any browser properties. To determine if using Selenium can cause significant differences in the results of a measurement study we compare Selenium-driven and non-Selenium-driven versions of desktop Chrome and of desktop Firefox.

We find that Selenium-driven Firefox experiences far fewer third-party (84% fewer) and third-party tracking-and-advertising (86%) requests than non-Selenium-driven Firefox (Figure 15). We also observe a significant difference in API accesses to the Automation and Storage categories (Figure 16). Selenium-driven Firefox sets `navigator.webdriver` to indicate it is automated [72]. We find more accesses to this property on Selenium-driven Firefox, which suggests that websites may be checking and not deploying ads (i.e., making third-party requests) if it is set.

Selenium-driven Chrome experiences significantly more third-party requests (20% more) than non-Selenium-driven Chrome. Selenium-driven Chrome also has significantly fewer accesses for the Automation categories. When Selenium is driving Chrome, it adds a property to the document object that indicates automation: `document.$cdc_asdjflasutopfhvcZLmcfl_` [28]. We inspected some scripts performing accesses to the Automation category APIs and found logic specifically to detect Selenium (e.g., see Figure 11).

Result 1b: Selenium can measurably affect the ecological validity of a web measurement study due to websites' bot detection efforts.

4.1.3 Difference in Desktop and Mobile Content

Desktop browsers generally use a larger screen size than mobile browsers and thus may show more content. This can confound comparisons of the amount of requests or API accesses between desktop and mobile browsers.

Q1c: How can content be equalized between desktop and mobile browsers, and what effect does this have on web-privacy measurements?

Equalization of content is difficult because websites may dynamically increase the amount of content they show. This is exhibited in the two different kinds of scrolling behavior websites have: *dynamic-scrolling*, where one can keep scrolling and new content is loaded, and *static-scrolling*, meaning that the page has a fixed bottom. We identify scrolling behavior by programmatically scrolling down the page and observing whether `document.body.scrollHeight` changes.

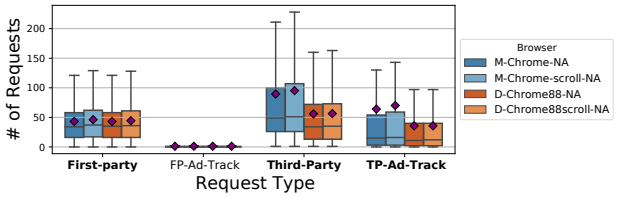

Fig. 2. Effect of scrolling on number of requests for static-scrolling sites. Bolded request types indicate significant differences.

The difficulty of equalization is exacerbated because websites often have a different layout on mobile browsers than on desktop browsers. For example, the desktop version of *youtube.com* renders as a two-dimensional grid of "suggested" videos. On a mobile browser a single column of videos is shown, with ads interspersed. In this case, does equalization mean that scrolling should be performed so that the mobile browser shows the same number of videos as the desktop browser? In small-scale experiments on *youtube.com* and other video sites we scrolled down on the mobile browser so that the same number of videos was shown as on the desktop browser and found little correlation between that form of equalization and new requests made on the mobile browser; some sites made more requests while others did not, and the new requests themselves were often unrelated to the content, e.g., periodic logging requests.

We suggest a different standard: Each browser should show in its viewport all the content fetched as a result of the top-level HTTP request. This is more difficult on dynamic-scrolling sites, as new content will continue to be loaded as the page is scrolled down. For these sites, we stop scrolling at what was the bottom of the page before new content was loaded (the "original bottom"). For static-scrolling websites, there is a fixed amount of content, which allows for content-based equalization. Static-scrolling sites make up about 70% of the sites in our crawl. We measure the effects of equalization by scrolling to the fixed bottom of static-scrolling and original bottom of dynamic-scrolling sites.

We compare request and API access behavior of mobile and desktop Chrome, with and without scrolling. For static-scrolling sites (Figure 2) we find that scrolling does increase content; mobile and desktop Chrome with scrolling have significantly more requests (6% on average) than their non-scrolling counterparts for every request type except for first-party tracking and advertising. For dynamic-scrolling sites we observe similar behavior (Figure 17). In both cases the increase in each

category of requests is proportional; both mobile and desktop Chrome with scrolling increase by a similar amount, with average increases within 1% of each other. Trends and conclusions in results will not change with this equalization; for example, mobile Chrome still has fewer first-party requests than desktop Chrome. Thus, the results we present (Section 4.2), do not use equalization via scrolling.

Result 1c: Equalization via scrolling results in proportional changes to mobile and desktop website behavior and thus does not affect conclusions about differences in website behavior.

4.2 Comparison of Browser Conditions

In this section we answer the following two sets of research questions. For Chrome and Firefox (*mainstream browsers*), which hold the largest market share of our browsers [82]: **Q2a**: How do mainstream mobile and desktop browsers compare in terms of tracking? We also compare a set of browsers that claim to include privacy-enhancing features (*privacy-focused* browsers): **Q2b**: How effective are privacy-focused browsers at blocking tracking and advertising? **Q2c**: Do tracking protections afforded by privacy-focused browsers differ in effectiveness between their mobile and desktop versions? **Q2d**: What are the strengths and weaknesses of individual privacy-focused browsers? Finally, considering both mainstream and privacy-focused browsers, we ask: **Q2e**: How does location affect tracking behavior?

4.2.1 Mainstream Mobile and Desktop Browsers

Q2a: How do mainstream mobile and desktop browsers compare in terms of tracking?

Previous work examined mobile and desktop Firefox [45, 106] but not mobile and desktop Chrome, which together hold 66% of total mobile and desktop market share. We compare mobile Chrome and mobile Firefox (termed *Mobile*) to desktop Chrome and desktop Firefox (termed *Desktop*) in terms of first-party and third-party requests, third-party tracking-and-advertising entities, API accesses, and potential fingerprinting.

Tracking and Advertising Requests. Figure 3 shows the differences between the number of mobile and desktop requests. The difference between the distributions for every request type is significant. For Firefox, the difference is particularly large for first-party

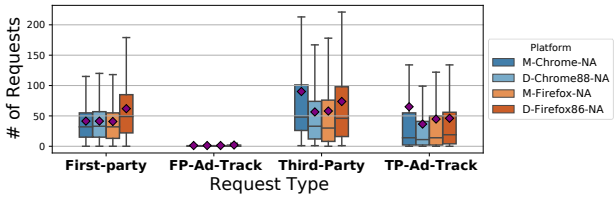

Fig. 3. Number of requests on mobile and desktop browsers.

(53% more on desktop) and third-party (55%) requests. For Chrome, the differences are largest for third-party (45% more on mobile) and third-party tracking-and-advertising (27%) requests. The difference in the number of third-party tracking-and-advertising requests experienced by a user on mobile or desktop depends on the browser: mobile Chrome experiences more requests than desktop Chrome, but desktop Firefox experiences more requests than mobile Firefox.

Since many domains have the same owner, we use an updated fork of the webXray domain owner list [61] to trace tracking-and-advertising requests to domain owners (see Section 3.3). We then aggregate tracking findings at the level of the parent entity to reveal the full reach of an entity. While we do find a significant difference in the distributions of entity counts, the median number of entities requested by the mobile and desktop versions of both browsers is the same (Figure 18). Likewise, the top four most prevalent entities are the same: Google, Facebook, Adobe Systems, and Microsoft.

Comparing the mobile browsers jointly against the desktop browsers, we find that none of the top 100 entities (by prevalence) is exclusive to mobile or desktop. Of the 5326 third-party tracking-and-advertising entities we recorded, just 357 are mobile-specific and 523 are desktop-specific. While prior work has pointed out this long tail of tracking [43, 106], we also report on their prevalence: The most prevalent mobile-specific entity, *ymetrica1.com* accounts for just 0.16% of the mobile requests in our crawl. Likewise, the most prevalent desktop-specific entity, *netshelter.net* accounts for just 0.01% of desktop requests. Collectively, all of the mobile-specific and desktop-specific entities account for just 0.45% requests in our crawl. The third-party tracking-and-advertising ecosystem on mobile and desktop hence seems more homogeneous than suggested by prior work.

Result 2a-1: The mobile and desktop versions of Chrome and Firefox experience significantly different amounts of third-party tracking-and-advertising requests, but the ecosystem of third-party entities is more homogeneous than previously reported.

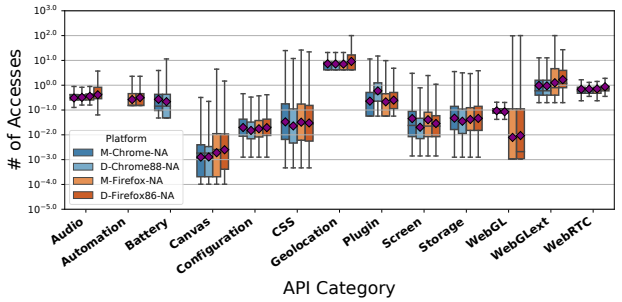

Fig. 4. APIs accessed on mobile and desktop browsers. The y-axis is the averaged number of API accesses on a page, normalized between mobile and desktop, and log-scaled for clarity.

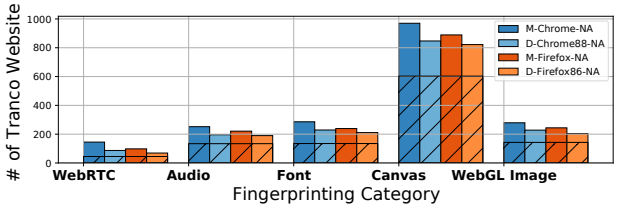

Fig. 5. Number of sites performing fingerprinting on mobile and desktop Chrome and Firefox. The cross-hatched area indicates that the fingerprinting category is present on all browsers.

JavaScript API Accesses and Fingerprinting. Measuring the behavior of a website in terms of its access to browser APIs is critical for quantifying the extent of tracking. Figure 4 shows the number of API accesses on mobile and desktop Chrome and Firefox for each sensitive JavaScript API category (Section 2).

Mobile Chrome and Firefox both experience significantly more API accesses to the Screen (54% more, on average) and Storage (42%) categories. Mobile Chrome also experiences significantly more accesses to the CSS category (57%). The above differences are largely due to more activity on mobile for *googlesyndication.com* and *googletagservices.com*, which respectively are 10th and 11th most prevalent third-party tracking-and-advertising domains [38]. For *googletagservices.com* and *googlesyndication.com* we see a greater number of API accesses (401% and 155%, respectively) on the mobile versions. The API access behavior of these two domains jointly account for 35% of total API accesses of the top 100 third-party tracking-and-advertising domains on mobile, underscoring the outsized effect of highly prevalent entities on the browsing experience of users. Desktop Chrome and Firefox experience significantly more API accesses to the Plugin category (50% more). This is unsurprising; the last remaining 3d-party plugin was Flash and it was not available on Android by default after Android 4.1 [89].

The APIs in the above categories are well-known vectors for fingerprinting (see Appendix A.1). For example, the APIs in the Screen category allow a website to determine a browsers' screen size, which has shown to result in differences in DOM and JavaScript content shown by the Alexa Top 100 websites [106]. However, these APIs have legitimate uses outside of fingerprinting. To reduce false positives, we apply several categories of fingerprinting heuristics (see Appendix A.1) and for each browser count the number of websites identified as performing fingerprinting in each category.

We find significant differences for the WebRTC (21% more on mobile, on average), Audio (10%), WebGL image (6%), and Canvas (4%) categories (Figure 5). To investigate further, we selected ten websites with large differences in fingerprinting behavior and manually reviewed the scripts loaded by those sites. We looked for characteristics of fingerprinting scripts [101] and found several mobile-specific scripts. For example, *antifraudjs.friends2follow.com* serves mobile browsers scripts that we detect performing WebGL image fingerprinting (confirmed by the domain's privacy policy which states that the script will perform browser fingerprinting [49]). We speculate the fingerprinting script is conditionally loaded by websites based on the platform. Additionally, scripts from *googletagservices.com* and *googlesyndication.com*, which play a large role in the greater number of API accesses to sensitive API categories, are recognized as performing fingerprinting [38].

Result 2a-2: Mobile Chrome and Firefox experience significantly more API accesses than their desktop counterparts for key APIs implicated in fingerprinting. These browsers also make requests to more pages whose behavior is indicative of fingerprinting.

4.2.2 Effectiveness of Privacy-focused Browsers

We next compare privacy-focused browsers—Ghostery, Firefox Focus, DuckDuckGo, Tor, and Brave—to Chrome and Firefox. Here we discuss only *mobile* browsers; an evaluation of their desktop counterparts, omitted for space, reaches the same conclusion.

Q2b: How effective are privacy-focused browsers at blocking tracking and advertising?

Tracking and Advertising Requests. When comparing the distributions of requests between privacy-focused (termed *Privacy*) browsers and Chrome and Firefox (termed *Mainstream*) we find that Privacy browsers overall experience significantly fewer third-party requests (Figure 19); most importantly, Main-

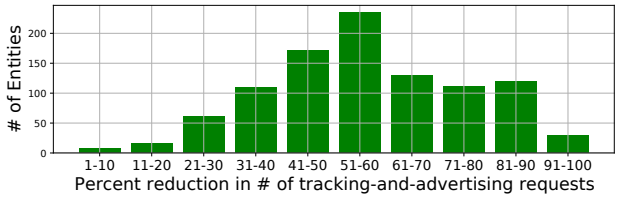

Fig. 6. Distribution of reductions in third-party tracking-and-advertising requests per entity achieved by privacy-focused mobile browsers, as compared to mainstream browsers. For example, we find that 235 entities have a 51-60% reduction in third-party tracking-and-advertising requests.

stream browsers experience 70% more third-party tracking-and-advertising requests on average. Privacy browsers also experience significantly fewer third-party tracking-and-advertising entities (Figure 21); Mainstream browsers experience double the number.

We examine which entities are responsible for the third-party tracking-and-advertising requests in each group. For both groups the top four most prevalent entities were Google, Facebook, Adobe Systems, and Microsoft. Privacy-focused browsers do not reduce the number of requests from these entities uniformly: As a group, they block about 60% of the tracking-and-advertising requests associated with Google, 66% for Facebook, 56% for Adobe Systems, and 67% for Microsoft. For the top 1000 entities find that the reduction in tracking is widely distributed, with a median reduction of 55% (Figure 6).

As a consequence, as we consider less prevalent third-party entities, we find less effective blocking behavior. For example, there is only a 22% reduction in tracking-and-advertising requests from Tencent. In the top 1000, there are 84 entities where Privacy browsers have a reduction of 30% less. For seven of those entities the Privacy browsers do not differ significantly from the Mainstream browsers in terms of third-party tracking-and-advertising.

These results suggest that the blocking lists used by privacy-focused mobile browsers are only consistently effective at reducing tracking-and-advertising requests from prevalent sources. This may be because (1) blocking lists are often community-generated and thus cannot cover all entities and (2) as blocking lists add rules for lower-ranked entities, the performance for querying the list degrades [93]. Therefore, rules for less-prevalent entities are excluded because of their performance cost.

JavaScript API Accesses and Fingerprinting. Since Privacy browsers are effective at reducing requests to prevalent third-party tracking-and-advertising entities, we examine how this correlates with API accesses. We find significant differences between the Privacy and Mainstream browser groups for every API access category besides Automation (Figure 20). The differences are large for several sensitive API categories: Screen (80% reduction), Storage (77%), and Configuration (46%). By blocking some of the most prevalent trackers, privacy-focused browsers also remove many potentially fingerprinting API accesses. Using our fingerprinting heuristics we confirm that privacy-focused browsers experience fingerprinting on significantly fewer websites for all categories, most notably WebGL Image (48% reduction) and Canvas (44%) (Figure 26).

Result 2b: Privacy-focused browsers experience a significant reduction in accesses to sensitive APIs as compared to mainstream browsers, and a reduction in third-party tracking-and-advertising requests. However, these reductions occur predominantly for prevalent tracking-and-advertising entities, suggesting a need for prioritizing block list comprehensive over performance for privacy.

4.2.3 Comparing Privacy-focused Mobile and Desktop Browsers

In Section 4.2.1, we showed that mainstream mobile browsers experience significantly different amounts third-party tracking-and-advertising requests than their desktop counterparts. Now, we perform the same comparison for privacy-focused browsers.

Q2c: Do tracking protections afforded by privacy-focused browsers differ in effectiveness between their mobile and desktop versions?

We investigate this question through a comparison of two privacy-focused browsers that have a desktop and a mobile version: Brave and Tor. We find that the mobile versions of the browsers have significantly fewer first-party (6% less) requests than their desktop counterparts (Figure 22). While their distributions of third-party and third-party tracking-and-advertising requests differ significantly, the median number of requests is the same. Comparing the entities responsible for third-party tracking-and-advertising requests, we find that the distribution of entity counts differ significantly for both Brave and Tor (Figure 24). However, the median number of entities is the same for both mobile and desktop, as are the five most prevalent entities: Google, Facebook, Adobe Systems, Microsoft, and Amazon.

Of the top 1000 entities, seven are visited only by privacy-focused mobile browsers and six only by their

desktop counterparts. These entities only receive 0.37% and 0.04% of requests, respectively, in the whole crawl. Investigation of the API access behavior of these entities did not show any notable differences in mobile or desktop behavior. We conclude that largely the same actors are performing tracking and advertising on privacy-focused desktop and mobile browsers.

In terms of API accesses, we find significant differences for all API accesses categories except Battery and WebRTC (Figure 23), with Screen (200% more on mobile) and Storage (100%) showing the greatest difference. We earlier reported similar differences between mainstream mobile and desktop browsers. However, on average the differences are 2.5 times larger in the privacy-focused comparison. This indicates that the API access differences intrinsic to mainstream mobile and desktop browsers are present for privacy-focused browsers to a greater degree.

We reported in Section 4.2.2 that privacy-focused browsers' tracking protections were effective at reducing third-party tracking-and-advertising. It is clear that this effect is *proportional* to the amount of tracking and advertising that mobile and desktop experience, respectively. Privacy-focused mobile and privacy-focused desktop browsers have a similar level of reduction of tracking-and-advertising requests. The difference between privacy-focused mobile and desktop browsers parallels the fewer requests and increased API accesses that mainstream mobile browsers have in comparison to mainstream desktop browsers.

Result 2c: Mobile and desktop privacy-focused browsers experience a similar amount of third-party tracking-and-advertising requests. Like mobile and desktop mainstream browsers, mobile privacy-focused browsers access sensitive APIs more often than their desktop counterparts.

4.2.4 Comparing Individual Privacy-focused Browsers

We reported in Section 4.2.2 that privacy-focused browsers are effective at reducing third-party tracking-and-advertising. However, we find variance in how well they perform, which leads to the following question.

Q2d: What are the relative strengths and weaknesses of individual privacy-focused browsers?

The term "privacy browser," commonly used on mobile app marketplaces, is overloaded, as different such browsers have different features. We categorize claimed features as follows: (1) ad blocking, (2) tracker blocking, (3) tracking protection, and (4) fingerprinting protection. (1) and (2) are well-defined in that they respectively imply reduction in third-party advertising and tracking requests. It is less clear what (3) implies, and (4) is counterintuitive because of what Eckersley called the "Paradox of Fingerprintable Privacy Enhancing Technologies" [42]: the presence of ad and tracker blocking can make a browser more fingerprintable. Thus, (4) may actually imply *no reduction* in third-party tracking-and-advertising requests and so can be in conflict with (1) and (2).

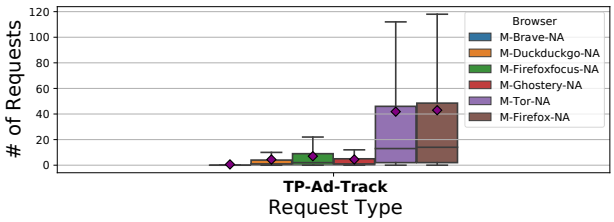

Fig. 7. Privacy-focused browsers compared in terms of third-party tracking-and-advertising requests.

On the Google Play Store (as of June 2021): Brave claims (1) and (3) [1]. Tor claims (2) and (4) [6]. Firefox Focus claims (1) and (2) [4]. DuckDuckGo claims (2) [2]. and Ghostery claims (1) and (2) [5]. While not primarily marketed as privacy-focused, we also include Firefox in this comparison as it too includes built-in tracking protection and claims (2) [3].

We investigate the request behavior of each browser (Figure 7). Brave is the most effective at reducing third-party tracking-and-advertising requests; its claim to (1) is supported. The other privacy-focused browsers vary significantly in terms of reducing third-party tracking-and-advertising requests. Firefox Focus and Ghostery differ significantly in how much they reduce third-party tracking-and-advertising requests, but the performance of both supports their claims, (1) and (2). Firefox is least effective at reducing third-party tracking-and-advertising requests. This is unsurprising, as Firefox's enabled-by-default "standard" tracking protection "blocks fewer trackers" [79]. It errs on the side of caution in what it blocks, focusing on restricting tracking cookies, social-media trackers, and fingerprinting scripts; it does not block other content (loaded ads, videos) that may perform tracking [78, 105]. Thus, while Firefox claims (2), its blocking of tracking and advertising is limited with its default settings.

Tor also claims (2), but it is the second least effective at reducing third-party tracking-and-advertising requests. What may be unclear to users is that this occurs by design: Tor is developed with the goal that "all Tor

users should have the exact same fingerprint" [84]. As a result, the Tor browser does not include any ad-blocking plugins and discourages users from adding such plugins as they would make the browser more unique [97], supporting the claim to (4). As discussed previously, (2) and (4) are at odds; Tor's claim to (2) could be clarified or removed to reduce potential confusion.

When it comes to the number of API accesses each browser experiences, the browsers also vary significantly (Figure 25): On average over all API categories, Brave has the fewest API accesses; 25% less than the group average. Firefox performs the worst overall, with 41% more accesses than the group average. Tor again performs the second-worst, with 35% more API accesses than the group average. As a result of blocking fewer third-party tracking-and-advertising requests, Tor has significantly more API accesses for several categories: Plugin (200% more), Screen (170%), Storage (77%), and Configuration (44%). However, when it comes to sensor APIs (Audio, Battery, and Geolocation), Tor has the least accesses. This is because Tor disables these APIs in order to reduce its fingerprinting surface [96].

The variation in the behavior of these browsers indicates a need for more clarity on app marketplaces as to what a "privacy browser" delivers. If a user's goal is to reduce ads and trackers on the web page (e.g., to receive the least number of tracking-and-advertising requests) then Brave is the clear winner. However, if the goal is to reduce access to sensor APIs that can be used for fingerprinting, Tor is the more effective option.

Result 2d: Privacy-focused mobile browsers vary significantly in effectiveness and, in some cases, deviate from what they claim to provide. This suggests a need for more clarity on app marketplaces as to the privacy implications of each browser.

4.2.5 Effect of Location on Tracking

We compare the amount of tracking on mainstream and privacy-focused browsers in North America (NA) and Asia (AS) to answer the following question.

Q2e: How does location affect tracking behavior?

The number of third-party tracking-and-advertising requests made by mainstream mobile and desktop browsers in NA is significantly different (4% more on mobile) than the number of requests made by those browsers in AS (Figures 27–28). Comparing privacy-focused browsers in NA to their counterparts in AS, we find that while the distributions of third-party tracking-and-advertising differ significantly, the median number of requests remains the same (Figures 29–30). This suggests that there is more third-party tracking-and-advertising in NA than AS and that privacy-focused browsers are effective at reducing third-party tracking-and-advertising in both locations.

Next, considering third-party tracking-and-advertising entities, we again find that while the distributions of entities differ significantly between NA and AS, the median number of entities remains the same. Further exploring country-specific tracking behavior, we find that the top four third-party tracking-and-advertising entities are the same in NA and AS: Google, Facebook, Adobe Systems, and Microsoft. Overall, 1091 entities (20%) are observed only in NA and 539 entities (10%) only in AS. However, these entities account for only 1.14% of the third-party tracking-and-advertising requests in NA and 2.32% of those in AS. Apart from differences in lower-ranking entities, we observe a country-specific tracking entity: AT&T, which ranks 5th on mobile and 7th on desktop in AS but outside the top 50 in NA. Specifically, the domain **.adnxs.com*, owned by AT&T, is a country-specific ad-and-tracking domain in AS. The observation of country-specific entities conforms with previous work [52].

Result 2e: Mainstream browsers experience significantly more third-party tracking-and-advertising requests in NA. Privacy-focused browsers are effective at reducing this tracking-and-advertising in both NA and AS. While 20% percent of NA entities and 10% of AS entities performing tracking and advertising differ based on browser location, they account for a small fraction of requests, and the most prominent entities are the same.

4.3 Other Interesting Findings

Ad-blocker Circumvention. Websites have tried to evade ad-blocking by using URLs not black-listed by ad-blockers. Our study found evidence that some companies have attempted to evade block lists by registering multiple seemingly unrelated domains.

We say that Domain A is an *evading domain* of a known tracking-and-advertising Domain B if A is not a known tracking-and-advertising domain but A and B share the same URL paths and HTTPS certificate. We find that B is often used to serve the same tracking-and-advertising content blocked by blocklists targeting A (but missing B).

Domain A shares a URL path with Domain B if it has at least one URL path fully matching a tracking-and-advertising request on Domain B, which implies that the two domains may use the same backend. Domain A shares a certificate with Domain B if the Subject Alternative Name or Common Name of domain A's certificate is also valid for Domain B or vice versa, which implies that A and B may belong to the same entity. For instance, we found that *s.adnxtr.com*, *s.cdnsynd.com*, and *s.xlgmedia.com* are evading domains of a known tracking domain, *s.tagsrvcs.com*. Those evading domains have the request URL path */2/4.69.1/main.js*, which is the same as the tracking request *s.tagsrvcs.com/2/4.69.1/main.js*. Additionally, all of them share the same HTTPS certificates.

Among the 73,142 (28,787 eTLD+1) domains that are not tracking-and-advertising domains in our crawl, we found 34,800 (20,217) domains that can be linked to at least one tracking-and-advertising domain using only the URL path. Furthermore, we found 2253 (1410) domains in our dataset that share the same HTTPS certificate with a tracking-and-advertising domain. By computing the intersection of these sets of domains, we identify 989 (681) evading domains.

While advanced ad-blockers may be able to use the above heuristic to identify and blacklist evading domains, the heuristic has the following false positives. We observe the prevalence of third-party websites/CDN hosting services that use the same certificate for all the hosted domains, conforming with previous findings [25]. For example, the Subject Alternative Names field of a tracking-and-advertising domain *cdn.bee7.com* includes *www.berkeley.edu*, a website of a US university, and *m-i-d.co.jp*, an online boutique shopping website in Japan. It is unlikely that they all belong to the same entity. Similarly, with respect to URL matching, a common path such as */jquery.min.js*, a popular JavaScript library, could flag benign domains. An interesting future direction is to study the impact of such certificate-sharing practices and the effectiveness of certificate or path matching on ad-blocking.

Similar Fingerprinting Code Snippets. In our study, we observed several scripts sharing similar fingerprinting code snippets. For example, out of the 1444 unique scripts that match at least one fingerprinting category, 277 (14%) contain all the magic strings of *fingerprintjs2*, a popular open-source library [101]. The magic strings *mmmmmmmmmmlli* (m takes up maximum width, lli adds entropy) and *Cwm fjordbank glyphs vext quiz* (English-language pangram) used for canvas fingerprinting appear in 250 and 319 of the scripts, respectively. 207 scripts use both the magic strings, and 162 of them directly embed the word *fingerprintjs*. Since magic strings are deliberately chosen to enable effective canvas fingerprinting, it is reasonable that they are left unchanged even if an entity would like to develop its own fingerprinting script. This could serve as a lightweight method to detect fingerprinting attempts.

Magic-string matching gives us a lower bound on the fraction of reused fingerprinting code snippets. However, it is challenging to determine the prevalence of code reuse and similarity for JavaScript, because most scripts are minified or obfuscated and may have runtime-resolved objects. Further analysis of code similarity is left for future work.

5 Related Work

In this section, we discuss related work on web tracking studies for both desktop and mobile platforms and compare our infrastructure with others.

5.1 Metrics to Measure Tracking

Tracking has evolved from stateful and cookie-based to stateless and fingerprinting-based, most recently leveraging hardware characteristics for fingerprinting, such as WebGL [26, 57, 69] and mobile sensors [14, 20, 33, 35, 108]. Unlike some earlier work [43, 44, 60, 86], our work did not analyze cookies. Also, similar to recent work, we analyze web requests to known tracking domains as a main metric for tracking [17, 43, 85, 106]. We additionally used webXray [61] data to understand tracking at the level of entities. Also similar to recent work, our second metric to measure tracking is the number of JavaScript API calls that could be used for fingerprinting [106]. We instrument a wider collection of sensitive APIs than previous work in order to increase the coverage of potential fingerprinting behaviors, such as WebGL, fetch and Selenium automation APIs. A final metric that we use to estimate tracking is fingerprinting heuristics. We adopt and refine some of the definitions used in literature to reduce false positives [7, 8, 33, 43, 57, 83].

Next, we expand on our design decisions of metrics that we did not use to measure tracking.

Cookie-based Tracking. Identifying identifier cookies used for tracking typically requires heuristics and is

Platform name	*Desktop browsers*	*Mobile browsers*	*JavaScript instrumentation*	*Stateful crawl*	*Browser driver*	*Scrolling*
FPDetective [8]	PhantomsJS, Chrome	-	Native WebKit code	○	S	○
OpenWPM [43]	Firefox	-	Browser-dependent plugins	●	S	○
OpenWPM-mobile [33]	Imitate mobile Firefox on desktop Firefox		Browser-dependent plugins‡	○	S*	●
FourthParty [66]	Firefox (plugin)	-	Browser-dependent plugins	○	C	○
webXray [61, 62]	Chrome	-	-	○	S	○
Eubank et al. [45]	-	Firefox	-	●	C	○
WebCapsule [81]	Blink-based browsers†	-	Blink DevTool	○	C	○
WTPatrol [106]	Firefox	Firefox	Browser-dependent plugins‡	○	C	●
CRAWLIUM [67]	Firefox	- §	Browser-dependent plugins‡	○	S	○
OmniCrawl	Chrome, Firefox, Tor, Brave, ...	Chrome, Firefox, Tor, Brave, ...	Browser-independent JavaScript	●	C	●

Fig. 8. Comparison of web crawling platforms. ●: supports, ◑: partially supports, ○: does not support. † = Chromium, Microsoft Edge, Brave, and Opera. ‡ = Uses OpenWPM's instrumentation. § = Focuses on Android applications, not browsers. Browser driver: S = Selenium, C = non-Selenium custom driver. * = Based on a modified Selenium as described in Section 4.1.1.

not precise [43, 44, 60, 86]. Further, counting requests to known tracking domains is a reasonable approximation. Nonetheless, our measurement infrastructure is configured to store cookies, and therefore, can be used by future studies that want to analyze cookies.

Entropy-based Fingerprinting. To analyze JavaScript-based fingerprinting, researchers have also performed entropy-based [19, 26, 42, 50, 56, 57, 102] and clustering-based analysis [33], which require knowing the distribution of API return values or the intents of different scripts, respectively. Without access to such distributions, we cannot perform entropy-based analysis. Our code and dataset will be made publicly available to facilitate such analyses in the future.

Network-level Tracking. Researchers have demonstrated how to perform browser-independent tracking via DNS requests [55], TLS implementations [95], and IP addresses [68]. Also, because our work aims at getting measurements on the client side, we do not cover server-side HTTP headers fingerprinting [26, 50, 57], whose results cannot be observed on the client.

5.2 Tracking on Mobile Phones.

After researchers identified fingerprintable APIs unique to mobile browsers [14, 20, 27, 33–35, 65, 108], a few studies measured such tracking with real [45, 106] or emulated [33] mobile browsers. Ours is the most comprehensive to date, in (1) the number of browsers (18 mobile and 24 desktop); (2) the type of browsers (mainstream and privacy-focused); (3) the realism of the crawling infrastructure (emulated vs. on real phone, Selenium-based vs. directly driven via Appium); and (4) the number of fingerprinting APIs monitored.

Fingerprinting by Mobile-specific Sensors. Although studies have shown that some fingerprinting techniques do not work well on mobile devices [53, 57], recent work has proposed new fingerprinting techniques via mobile-specific sensor APIs, such as motion [20, 34, 35, 108], proximity [33], ambient-light sensors [14], and magnetometers [27, 65]. OmniCrawl instruments these APIs to measure their use in the wild, even though, except for motion, the APIs are by default disabled in modern browsers (our browsers). Our results show prevalent use of these APIs, in line with prior results, that is not entirely mitigated by privacy-focused browsers.

Tracking Measurements Using Real and Emulated Mobile Devices. Eubank et al. compare stateless tracking behaviors on real desktop and mobile devices (all running the Firefox browser) [45]. However, they did not consider other types of non-emulated browsers, and their measurement involved only 500 sites. Van Goethem et al. investigate the security of mobile-specific websites, using a modified desktop Chromium browser emulating a mobile browser [100]. Das et al.'s OpenWPM-Mobile is an emulated mobile browser running on desktop Firefox, whose fingerprint is almost identical to real mobile Firefox, except the Canvas and WebGL fingerprint [33]. Finally, Yang et al. performed a comparison of tracking on desktop and mobile Firefox on websites that have mobile-specific versions [106].

We compare a wide collection of non-emulated and emulated browsers (including OpenWPM-Mobile) and show that even though careful emulation improves eco-

logical validity, it is still insufficient for completely accurate results. Nonetheless, our results using Chrome and other browsers agree with prior findings using Firefox: desktop browsers typically visit more tracking and advertising third-parties than mobile browsers. We additionally show results on privacy-focused browsers.

5.3 Web Measurement Infrastructure

Figure 8 compares nine web crawling tools with OmniCrawl with respect to desktop and mobile browser implementation, JavaScript instrumentation, the support of stateful crawl, automation driver, and scrolling support. While existing tools require considerable effort to support different browsers on both mobile and desktop platforms, OmniCrawl is designed to reduce browser dependency, which is beneficial for synchronized cross-browser and cross-platform crawling.

5.4 Measurement Study Ecological Validity

Ahmad et al. qualitatively evaluated a set of crawling tools to determine their suitability for particular types of measurement studies [13]. They compare command-line tools like cURL to browser drivers like Selenium and the measurement platform OpenWPM and find that user-layer crawlers (ULC) like OpenWPM provide more accurate measurements of online tracking than simpler crawlers like cURL. OmniCrawl is a ULC and thus is suitable for online tracking measurement. We also examine the relative impacts of different crawling approaches, but we focus more narrowly and deeply on the large-scale effects of emulated browsers and browser drivers.

Vastel el al. measured the detection of automated crawlers within the Alexa top 10K sites and investigate the techniques to detect crawlers [103]. They found 291 websites that block crawlers and identify that 32% of these websites use fingerprinting to perform detection of crawlers. This finding is consistent with our finding that the behavior of websites can change when a Selenium-driven browser is used (Section 4.1.2). However, unlike Vastel el al. we also measure the ecological validity (in terms of differences in counts of requests and API accesses) of OpenWPM-Mobile, which focuses on faithfully simulating a real mobile browser (Section 4.1.1).

Several studies investigate the representativeness of automated crawls in comparison to real user behavior [52, 107] and suggest that automated crawlers might over-approximate the amount of third-party trackers a real user would experience. This is a limitation inherent to automated crawlers that may be ameliorated by future work on informing crawler behavior with real user behavior. Our evaluation focuses on the scope of validity of automated crawlers, real user behavior notwithstanding. To this extent, our comparison of baseline variation is similar to Zeber et al.'s [107]; we also reach the conclusion that automated crawler visiting sites at the same time receive similar content and thus can serve as a basis for comparison of other variables.

6 Conclusion

We build OmniCrawl, a scalable web crawling infrastructure capable of visiting websites with many browsers in parallel and on mobile phones and desktop computers. Using OmniCrawl, we perform a comprehensive analysis of web tracking on mobile, desktop, and privacy-focused browsers. We find that mobile browsers receive fewer tracking-and-advertising requests than desktop browsers, but also that the same entities are responsible for tracking and advertising on mobile and desktop browsers. Privacy-focused browsers are effective at reducing tracking-and-advertising requests, although they perform very differently from each other. Our results also suggest a trade-off between reducing tracking and advertising requests and being susceptible to fingerprinting. Our analysis also shows that common methodological choices, such as using an emulated mobile browser or a browser driver like Selenium, can easily impact the accuracy of web tracking measurement.

7 Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments and feedback. This work was supported in part by the National Science Foundation via grant CNS1704542 and the Ministry of Science and Technology of Taiwan (MOST 110-2628-E-002-002).

References

[1] Google Play Store page for Brave. https://play.google.com/store/apps/details?id=com.brave.browser, 2021.

[2] Google Play Store page for DuckDuckGo. https://play.google.com/store/apps/details?id=com.duckduckgo.

mobile.android, 2021.

[3] Google Play Store page for Firefox. https://play.google.com/store/apps/details?id=org.mozilla.firefox, 2021.

[4] Google Play Store page for Firefox Focus. https://play.google.com/store/apps/details?id=org.mozilla.focus, 2021.

[5] Google Play Store page for Ghostery. https://play.google.com/store/apps/details?id=com.ghostery.android.ghostery, 2021.

[6] Google Play Store page for Tor. https://play.google.com/store/apps/details?id=org.torproject.torbrowser, 2021.

[7] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proc. ACM CCS*, 2014.

[8] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel. FPDetective: dusting the web for fingerprinters. In *Proc. ACM CCS*, 2013.

[9] AdGuard. AdGuard base filter. https://raw.githubusercontent.com/AdguardTeam/FiltersRegistry/master/filters/filter_2_English/filter.txt, 2021.

[10] AdGuard. AdGuard chinese ads filter. https://raw.githubusercontent.com/AdguardTeam/FiltersRegistry/master/filters/filter_224_Chinese/filter.txt, 2021.

[11] AdGuard. AdGuard mobile ads filter. https://raw.githubusercontent.com/AdguardTeam/FiltersRegistry/master/filters/filter_11_Mobile/filter.txt, 2021.

[12] AdGuard. AdGuard tracking protection filter. https://raw.githubusercontent.com/AdguardTeam/FiltersRegistry/master/filters/filter_3_Spyware/filter.txt, 2021.

[13] S. S. Ahmad, M. D. Dar, M. F. Zaffar, N. Vallina-Rodriguez, and R. Nithyanand. Apophanies or epiphanies? How crawlers impact our understanding of the web. In *Proc. WWW*, 2020.

[14] M. Azizyan, I. Constandache, and R. Roy Choudhury. SurroundSense: mobile phone localization via ambience fingerprinting. In *Proc. MOBICOM*, 2009.

[15] S. Badle. Selenium—web browser automation. https://www.seleniumhq.org/, 2019.

[16] P. Barford, I. Canadi, D. Krushevskaja, Q. Ma, and S. Muthukrishnan. Adscape: harvesting and analyzing online display ads. In *Proc. WWW*, 2014.

[17] M. A. Bashir, S. Arshad, W. Robertson, and C. Wilson. Tracing information flows between ad exchanges using retargeted ads. In *Proc. USENIX*, 2016.

[18] BetaFish Inc. AdBlock. https://getadblock.com/, 2019.

[19] K. Boda, Á. M. Földes, G. G. Gulyás, and S. Imre. User tracking on the web via cross-browser fingerprinting. In *Proc. NordSec*, 2011.

[20] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh. Mobile device identification via sensor fingerprinting. *arXiv preprint arXiv:1408.1416*, 2014.

[21] T. Book and D. S. Wallach. An Empirical Study of Mobile Ad Targeting. *arXiv:1502.06577 [cs]*, 2015. arXiv: 1502.06577.

[22] Brave Software, Inc. Brave passes 15 million monthly active users and 5 million daily active users, showing 2.25x MAU growth in the past year. https://brave.com/15-million/, 2020.

[23] Brave Software, Inc. Secure, fast & private web browser with adblocker | Brave browser. https://brave.com/, 2020.

[24] BrowserLeaks. BrowserLeaks - web browser fingerprinting - browsing privacy. https://browserleaks.com/, 2020.

[25] F. Cangialosi, T. Chung, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson. Measurement and analysis of private key sharing in the https ecosystem. In *Proc. ACM CCS*, 2016.

[26] Y. Cao, S. Li, and E. Wijmans. (Cross-)browser fingerprinting via OS and hardware level features. In *Proc. NDSS*, 2017.

[27] Y. Cheng, X. Ji, J. Zhang, W. Xu, and Y.-C. Chen. DeMiCPU: Device fingerprinting with magnetic signals radiated by CPU. In *Proc. ACM CCS*, 2019.

[28] Chromium Project. Issue 3220: Websites can detect use of chromedriver or Selenium through the 'getPageCache' key - chromedriver. https://bugs.chromium.org/p/chromedriver/issues/detail?id=3220, 2019.

[29] Cliqz GmbH. Ghostery extension. https://www.ghostery.com/products/, 2019.

[30] Cliqz GmbH. Ghostery privacy browser. https://play.google.com/store/apps/details?id=com.ghostery.android.ghostery&hl=en, 2019.

[31] W. J. Conover and R. L. Iman. Rank Transformations as a Bridge Between Parametric and Nonparametric Statistics. *The American Statistician*, 1981.

[32] A. Cortesi, M. Hils, T. Kriechbaumer, and contributors. mitmproxy: A free and open source interactive HTTPS proxy. https://mitmproxy.org/, 2010–. [Version 4.0].

[33] A. Das, G. Acar, N. Borisov, and A. Pradeep. The web's sixth sense: A study of scripts accessing smartphone sensors. In *Proc. ACM CCS*, 2018.

[34] A. Das, N. Borisov, and M. Caesar. Tracking mobile web users through motion sensors: Attacks and defenses. In *Proc. NDSS*, 2016.

[35] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi. AccelPrint: Imperfections of accelerometers make smartphones trackable. In *Proc. NDSS*, 2014.

[36] Disconnect, Inc. Disconnect. https://disconnect.me, 2019.

[37] DuckDuckGo. DuckDuckGo app. https://duckduckgo.com/app, 2019.

[38] DuckDuckGo. Duckduckgo tracker radar. https://github.com/duckduckgo/tracker-radar, 2021.

[39] S. Dudoit and M. J. v. d. Laan. *Multiple Testing Procedures with Applications to Genomics*. Springer Series in Statistics. 2008.

[40] EasyList. EasyList without rules for adult sites. https://easylist-downloads.adblockplus.org/easylist_noadult.txt, 2021.

[41] EasyList. EasyPrivacy rules list. https://easylist.to/easylist/easyprivacy.txt, 2021.

[42] P. Eckersley. How unique is your web browser? In *Proc. PETS*, 2010.

[43] S. Englehardt and A. Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proc. ACM CCS*, 2016.

[44] S. Englehardt, D. Reisman, C. Eubank, P. Zimmerman, J. Mayer, A. Narayanan, and E. W. Felten. Cookies that give you away: The surveillance implications of web tracking. In *Proc. WWW*, 2015.

[45] C. Eubank, M. Melara, D. Perez-Botero, and A. Narayanan. Shining the floodlights on mobile web

tracking—A privacy survey. In *Proc. IEEE W2SP*, 2013.

[46] eyeo GmbH. AdBlock Plus. https://adblockplus.org/, 2019.

[47] Freedom House. Freedom in the world. https://freedomhouse.org/sites/default/files/2020-02/FH_FIW_Report_2018_Final.pdf, 2018.

[48] M. Friedman. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, 1937.

[49] Friends2Follow. Friends2follow privacy policy. https://friends2follow.com/privacy.html, 2021.

[50] A. Gómez-Boix, P. Laperdrix, and B. Baudry. Hiding in the crowd: an analysis of the effectiveness of browser fingerprinting at large scale. In *Proc. WWW*, 2018.

[51] S. Holm. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, 1979.

[52] X. Hu, G. Suaréz de Tangil, and N. Sastry. Multi-country study of third party trackers from real browser histories. In *Proc. IEEE European S & P*, 2020.

[53] T. Hupperich, D. Maiorca, M. Kührer, T. Holz, and G. Giacinto. On the robustness of mobile device fingerprinting: Can mobile users escape modern web-tracking mechanisms? In *Proc. ACSAC*, 2015.

[54] H. Jonker, B. Krumnow, and G. Vlot. Fingerprint surface-based detection of web bot detectors. In *Proc. ESORICS*, 2019.

[55] A. Klein and B. Pinkas. Dns cache-based user tracking. In *Proc. NDSS*, 2019.

[56] P. Laperdrix. FP central. https://fpcentral.tbb.torproject.org/about, 2016.

[57] P. Laperdrix, W. Rudametkin, and B. Baudry. Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. In *Proc. IEEE S & P*, 2016.

[58] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proc. NDSS*, 2019.

[59] M. Lecuyer, R. Spahn, Y. Spiliopolous, A. Chaintreau, R. Geambasu, and D. Hsu. Sunlight: Fine-grained Targeting Detection at Scale with Statistical Confidence. In *Proc. ACM CCS*, 2015.

[60] T.-C. Li, H. Hang, M. Faloutsos, and P. Efstathopoulos. Trackadvisor: Taking back browsing privacy from third-party trackers. In *Proc. ICPANM*, 2015.

[61] T. Libert. Exposing the hidden web: An analysis of third-party HTTP requests on 1 million websites. *International Journal of Communication*, 2015.

[62] T. Libert. webxray. https://webxray.org/, 2019.

[63] M. Lécuyer, G. Ducoffe, F. Lan, A. Papancea, T. Petsios, R. Spahn, A. Chaintreau, and R. Geambasu. XRay: Enhancing the Web's Transparency with Differential Correlation. In *Proc. USENIX*, 2014.

[64] H. B. Mann and D. R. Whitney. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, 1947.

[65] N. Matyunin, Y. Wang, T. Arul, K. Kullmann, J. Szefer, and S. Katzenbeisser. Magneticspy: Exploiting magnetometer in mobile devices for website and application fingerprinting. In *Proc. ACM WPES*, 2019.

[66] J. R. Mayer and J. C. Mitchell. Third-party web tracking: Policy and technology. In *Proc. IEEE S & P*, 2012.

[67] G. Merzdovnik, M. Huber, D. Buhov, N. Nikiforakis, S. Neuner, M. Schmiedecker, and E. Weippl. Block me if you can: A large-scale study of tracker-blocking tools. In *Proc. IEEE European S & P*, 2017.

[68] V. Mishra, P. Laperdrix, A. Vastel, W. Rudametkin, R. Rouvoy, and M. Lopatka. Don't count me out: On the relevance of IP address in the tracking ecosystem. In *Proc. WWW*, 2020.

[69] K. Mowery and H. Shacham. Pixel perfect: Fingerprinting canvas in HTML5. In *Proc. IEEE W2SP*, 2012.

[70] Mozilla Foundation. DeviceMotionEvent. https://developer.mozilla.org/en-US/docs/Web/API/DeviceMotionEvent.

[71] Mozilla Foundation. Firefox 56 release notes. https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Releases/56#plugins, 2017.

[72] Mozilla Foundation. Navigator.webdriver. https://developer.mozilla.org/en-US/docs/Web/API/Navigator/webdriver, 2018.

[73] Mozilla Foundation. Light api browser compatibility table. https://developer.mozilla.org/en-US/docs/Web/API/Ambient_Light_Events#Browser_compatibility, 2019.

[74] Mozilla Foundation. Magnetometer API browser compatibility table. https://developer.mozilla.org/en-US/docs/Web/API/Magnetometer#Browser_Compatibility, 2019.

[75] Mozilla Foundation. Proximity api browser compatibility table. https://developer.mozilla.org/en-US/docs/Web/API/Proximity_Events#Browser_compatibility, 2019.

[76] Mozilla Foundation. What is Firefox Focus? https://support.mozilla.org/en-US/kb/focus, 2019.

[77] Mozilla Foundation. Public suffix list. https://publicsuffix.org/, 2020.

[78] Mozilla Foundation. Enhanced tracking protection in firefox for android. https://support.mozilla.org/en-US/kb/enhanced-tracking-protection-firefox-android#w_see-what-is-protectedblocked-on-a-website, 2021.

[79] Mozilla Foundation. Firefox for Android version 86.1.1. https://github.com/mozilla-mobile/fenix/releases/tag/v86.1.1, 2021.

[80] S. Nath. MAdScope: Characterizing Mobile In-App Targeted Ads. In *Proc. MobiSys*, 2015.

[81] C. Neasbitt, B. Li, R. Perdisci, L. Lu, K. Singh, and K. Li. Webcapsule: Towards a lightweight forensic engine for web browsers. In *Proc. ACM CCS*, 2015.

[82] NetApplications.com. Browser market share. www.netmarketshare.com, 2021.

[83] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Proc. IEEE S & P*, 2013.

[84] Pierre Laperdrix. Browser fingerprinting: An introduction and the challenges ahead. https://blog.torproject.org/browser-fingerprinting-introduction-and-challenges-ahead?page=1, 2019.

[85] E. Pujol, O. Hohlfeld, and A. Feldmann. Annoyed users: Ads and ad-block usage in the wild. In *Proc. IMC*, 2015.

[86] F. Roesner, T. Kohno, and D. Wetherall. Detecting and defending against third-party tracking on the web. In *Proc.*

NSDI, 2012.

[87] S. Schelter and J. Kunegis. Tracking the trackers: A large-scale analysis of embedded web trackers. In *Proc. ICWSM*, 2016.

[88] M. Schwarz, F. Lackner, and D. Gruss. JavaScript template attacks: Automatically inferring host information for targeted exploits. In *Proc. NDSS*, 2019.

[89] G. Scott. Adobe: No Flash for You, Android 4.1. https://www.wired.com/2012/06/android-4-1-no-flash-you/.

[90] P. C. Sham and S. M. Purcell. Statistical power and significance testing in large-scale genetic studies. *Nature Reviews Genetics*, 2014.

[91] R. Shay, S. Komanduri, A. L. Durity, P. S. Huh, M. L. Mazurek, S. M. Segreti, B. Ur, L. Bauer, N. Christin, and L. F. Cranor. Designing Password Policies for Strength and Usability. *ACM Trans. Inf. Syst. Secur.*, 2016.

[92] R. Simpson. Mobile and tablet internet usage exceeds desktop for first time worldwide. https://gs.statcounter.com/press/mobile-and-tablet-internet-usage-exceeds-desktop-for-first-time-worldwide, 2016.

[93] B. Software. The mounting costs of stale ad-blocking rules. https://brave.com/the-mounting-cost-of-stale-ad-blocking-rules/, 2018.

[94] Statcounter. Mobile browser market share worldwide. https://gs.statcounter.com/browser-market-share/mobile/worldwide, 2020.

[95] E. Sy, C. Burkert, H. Federrath, and M. Fischer. Tracking users across the web via TLS session resumption. In *Proc. ACSAC*, 2018.

[96] Tor Project. The design and implementation of the Tor browser [draft]. https://2019.www.torproject.org/projects/torbrowser/design/#fingerprinting-linkability, 2018.

[97] Tor Project. Should I install a new add-on or extension in Tor Browser, like AdBlock Plus or uBlock Origin? https://support.torproject.org/tbb/tbb-14/, 2020.

[98] Tor Project. Tor Project | anonymity online. https://www.torproject.org/, 2020.

[99] Tor Project. Users – Tor metrics. https://metrics.torproject.org/userstats-relay-country.html?start=2020-05-01&end=2020-06-01&country=all&events=off, 2020.

[100] T. Van Goethem, V. Le Pochat, and W. Joosen. Mobile friendly or attacker friendly?: A large-scale security evaluation of mobile-first websites. In *Proc. ASIACCS*, 2019.

[101] V. Vasilyev. Fingerprint.js. https://github.com/Valve/fingerprintjs2, 2015.

[102] A. Vastel, P. Laperdrix, W. Rudametkin, and R. Rouvoy. FP-Scanner: the privacy implications of browser fingerprint inconsistencies. In *Proc. USENIX*, 2018.

[103] A. Vastel, W. Rudametkin, R. Rouvoy, and X. Blanc. FP-Crawlers: Studying the Resilience of Browser Fingerprinting to Block Crawlers. In *Proc. NDSS*, 2020.

[104] F. Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1945.

[105] M. Wood. Today's Firefox blocks third-party tracking cookies and cryptomining by default. https://blog.mozilla.org/en/products/firefox/todays-firefox-blocks-third-party-tracking-cookies-and-cryptomining-by-default/, 2021.

[106] Z. Yang and C. Yue. A comparative measurement study of web tracking on mobile and desktop environments. In *Proc. PETS*, 2020.

[107] D. Zeber, S. Bird, C. Oliveira, W. Rudametkin, I. Segall, F. Wollsén, and M. Lopatka. The representativeness of automated web crawls as a surrogate for human browsing. In *Proc. WWW*, 2020.

[108] J. Zhang, A. R. Beresford, and I. Sheret. SENSORID: Sensor calibration fingerprinting for smartphones. In *Proc. IEEE S & P*, 2019.

A Appendix

A.1 Fingerprinting Heuristics

We define five fingerprinting heuristics, all of which are at least as strict as those used in related work.

1. WebRTC: Invoke at least the four functions *createDataChannel*, *createOffer*, *setLocalDescription* and *onicecandidate* in the WebRTC APIs. [33, 43]
2. Audio: Satisfy both (1) Access at the 3 functions, *destination*, *startRendering* and *oncomplete*, in any AudioContext object. (2) Invoke *createOscillator* or *createDynamicsCompressor* [33, 43, 56].
3. Font: Satisfy both (1) Set $\geq$ 50 distinct fonts using *fontFamily*. (2) Detect element offset more than 50 times using *HTMLElement.offsetWidth* *HTMLElement.offsetHeight* or *CanvasRenderingContext2D.measureText* [8, 43, 83].
4. Canvas: Satisfy the following four conditions (1) Both width, height of the canvas must be $\geq$ 16 pixels. (2) Access *toDataURL* to retrieve image data. The image must not be JPEG. (3) Invoke *fillText* or *strokeText* and write at least 10 distinct characters. (4) Must not call *save* and *restore* APIs [7, 33, 43].
5. WebGL Canvas: Satisfy the following four conditions (1) Both width and height of the canvas must be larger than 16 pixels. (2) Access *toDataURL* and WebGL canvas to retrieve image data. The image must not be JPEG. (3) Attach at least one shader. (4) The image is not meaningful (through manual verification) [57, 69].

A.2 Baseline Variability in Website Behavior

We measured a baseline for how much websites vary in terms of requests, API accesses, and third-party tracking-and-advertising entities in a given visit. by including two identical copies of desktop Chrome running on the same machine in our crawl.

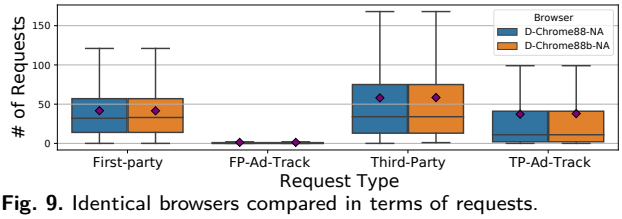

Fig. 9. Identical browsers compared in terms of requests.

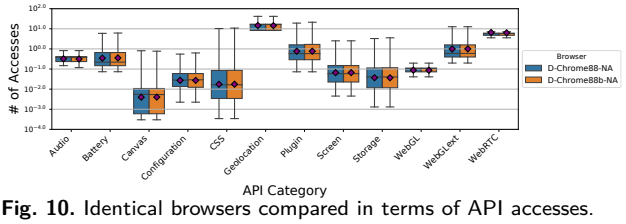

Fig. 10. Identical browsers compared in terms of API accesses.

First, looking at the four request types (Figure 9), we find that the two browsers measure nearly identically for all types, the difference between the two is not significant for any of the types. Next, we look at the third-party tracking-and-advertising entities and find that there isn't a significant difference in the number of entities. The entities are almost entirely the same, the top 1000 entities do not differ at all. Finally, considering API accesses (Figure 10) we likewise see that there isn't a significant difference for any of the API categories.

Result: The baseline level of variation in the sites we visit does not significantly affect our results.

A.3 Additional Figures

In this section we present additional figures referenced in the main text.

```
var r = ["__webdriver_evaluate", "
    __selenium_evaluate",...];
function s() {
    if (u(q)) return !0;
    var a = l(r, function(a) {
      return g.document[a] ? !0 : !1
    });
    if (a) return !0;
    ... }
```

Fig. 11. A JavaScript snippet for detection of Selenium from https://connect.facebook.net/en_US/fbevents.js.

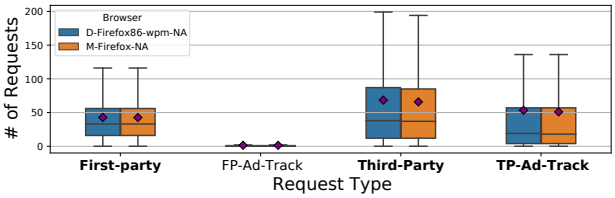

Fig. 12. OpenWPM-Mobile compared to mobile Firefox in terms of requests.

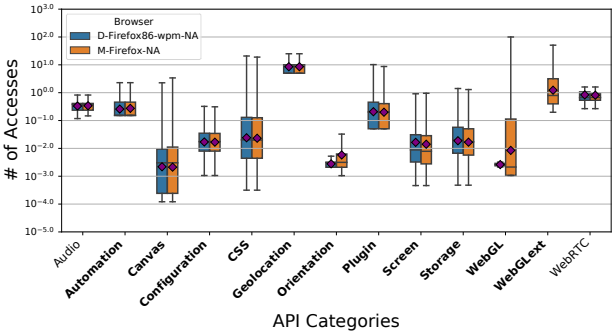

Fig. 13. OpenWPM-Mobile compared to mobile Firefox in terms of API accesses.

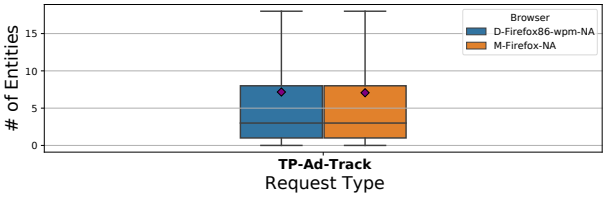

Fig. 14. OpenWPM-Mobile compared to mobile Firefox in terms of third-party tracking-and-advertising entities.

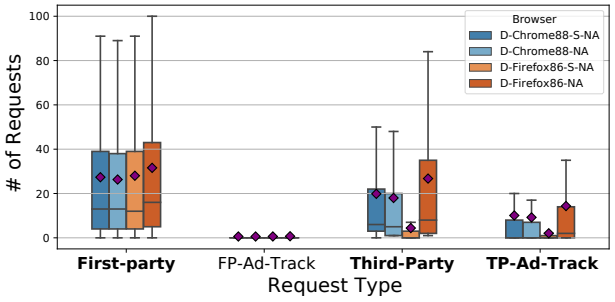

Fig. 15. Comparison of Selenium-driven vs non-Selenium-driven Browsers' Requests.

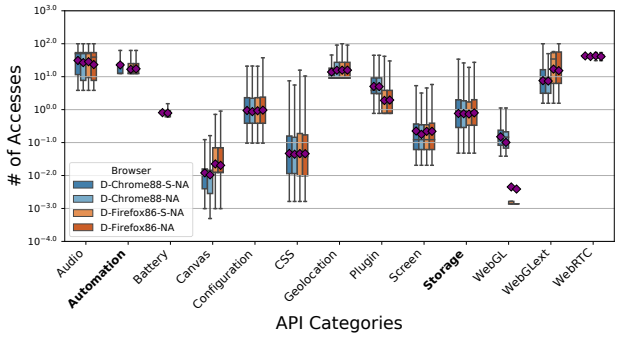

Fig. 16. Comparison of Selenium-driven vs non-Selenium-driven Browsers' API Accesses.

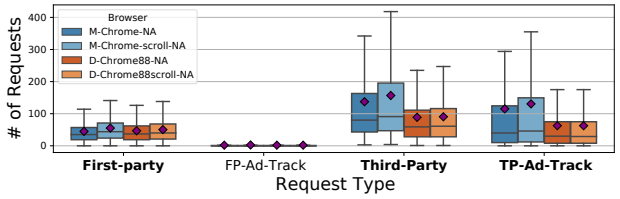

Fig. 17. Effect of scrolling on number of requests for dynamic-scrolling sites.

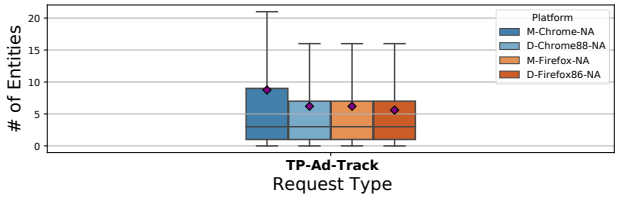

Fig. 18. Third-party tracking and advertising entities on mobile and desktop browsers.

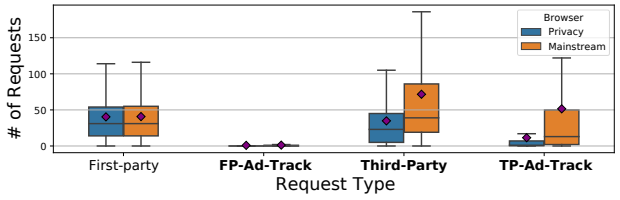

Fig. 19. Privacy-focused compared to mainstream mobile browsers in terms of requests.

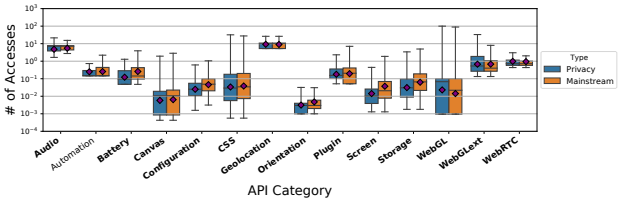

Fig. 20. Privacy-focused compared to mainstream mobile browsers in terms of API accesses.

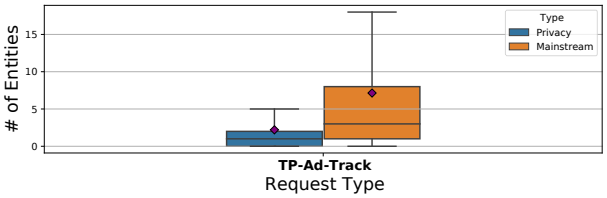

Fig. 21. Privacy-focused compared to mainstream mobile browsers in terms of third-party tracking-and-advertising entities.

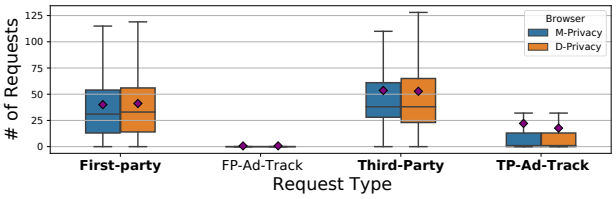

Fig. 22. Privacy-focused mobile and desktop browsers in terms of requests.

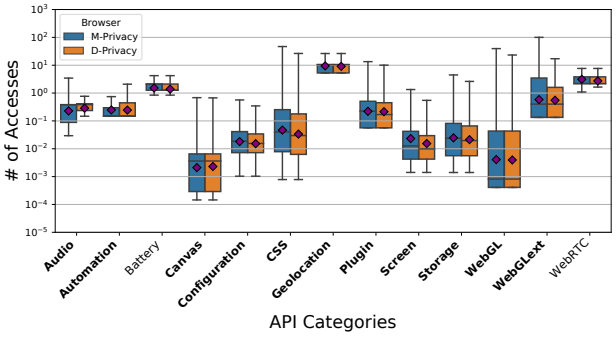

Fig. 23. Privacy-focused mobile and desktop browsers in terms of API accesses.

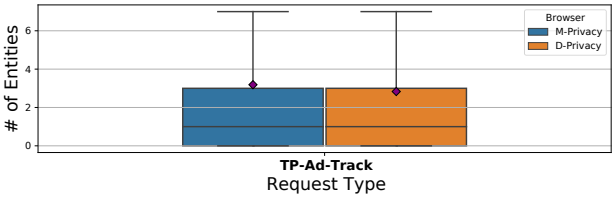

Fig. 24. Privacy-focused mobile and desktop browsers in terms of third-party tracking-and-advertising entities.

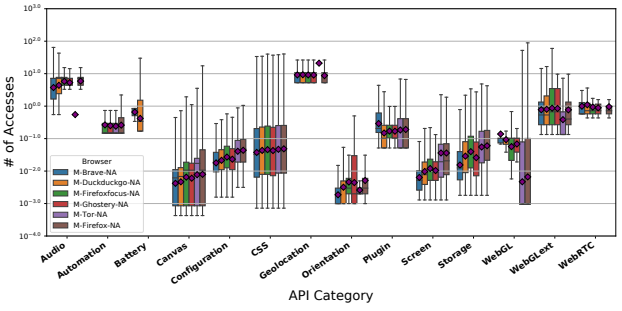

Fig. 25. Privacy-focused browsers compared by API category.

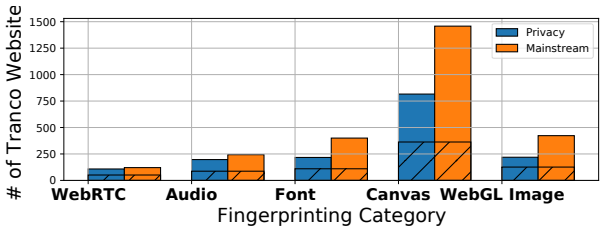

Fig. 26. Comparison of mobile privacy and mainstream browsers' fingerprinting behaviors.

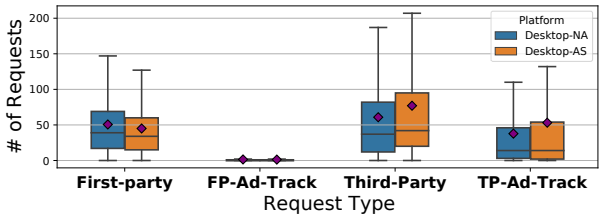

Fig. 27. Mainstream desktop browsers in Asia and NA in terms of requests.

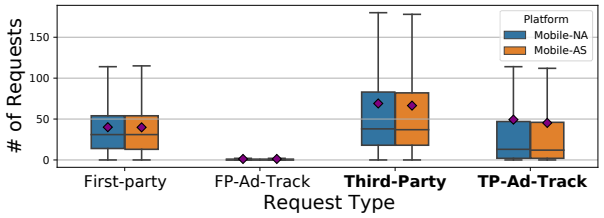

Fig. 28. Mainstream mobile browsers in Asia and NA in terms of requests.

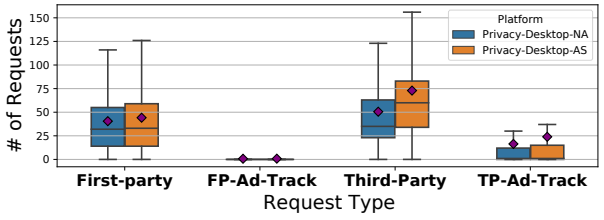

Fig. 29. Privacy-focused desktop browsers in Asia and NA in terms of requests.

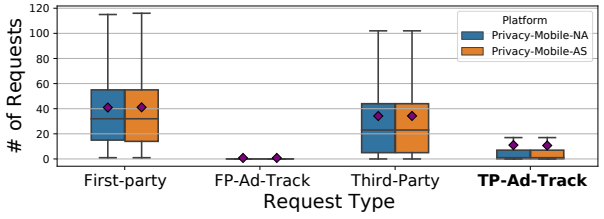

Fig. 30. Privacy-focused mobile browsers in Asia and NA in terms of requests.

A.4 Test Statistics

Test statistics were computed with a significance level of 0.05 unless otherwise stated. MW-2 and WSR-2 stand for two-sided Mann-Whitney U test and Wilcoxon signed rank test, respectively. Ms are the group medians ordered as the test is named (e.g., Mobile vs Desktop means the first median is for mobile, the second median is for desktop).

Test	*Result*
OpenWPM-Mobile vs mobile Firefox	
Requests	
First-party	Ms: $[33.0, 33.0]$ WSR-2 $W = 13631027.0$, $e = 0.18$, $n1 = n2 = 12460$, $p = 0.00$
FP-Ad-Track	Ms: $[0.0, 0.0]$ WSR-2 $W = 106169.0$, $e = 0.00$, $n1 = n2 = 12460$, $p = 0.79$
Third-Party	Ms: $[38.0, 37.0]$ WSR-2 $W = 19749497.0$, $e = 0.25$, $n1 = n2 = 12460$, $p = 0.00$
TP-Ad-Track	Ms: $[19.0, 18.0]$ WSR-2 $W = 8651805.0$, $e = 0.11$, $n1 = n2 = 12460$, $p = 0.00$
Third-party tracking-and-advertising entities	
TP-Ad-Track	Ms: $[3.0, 3.0]$ WSR-2 $W = 1429658.0$, $e = 0.02$, $n1 = n2 = 12460$, $p = 0.00$
API Accesses	
Audio	Ms: $[0.0, 0.0]$ WSR-2 $W = 1071.5$, $e = 0.00$, $n1 = n2 = 12460$, $p = 0.99$
Automation	Ms: $[0.0, 0.0]$ WSR-2 $W = 250110.0$, $e = 0.00$, $n1 = n2 = 12460$, $p = 0.00$
Canvas	Ms: $[0.0, 0.0]$ WSR-2 $W = 955759.5$, $e = 0.01$, $n1 = n2 = 12460$, $p = 0.03$
Configuration	Ms: $[32.0, 31.0]$ WSR-2 $W = 8525026.5$, $e = 0.11$, $n1 = n2 = 12460$, $p = 0.00$
CSS	Ms: $[39.0, 39.0]$ WSR-2 $W = 3783624.5$, $e = 0.05$, $n1 = n2 = 12460$, $p = 0.00$
Geolocation	Ms: $[0.0, 0.0]$ WSR-2 $W = 8385.5$, $e = 0.00$, $n1 = n2 = 12460$, $p = 0.03$
Orientation	Ms: $[0.0, 0.0]$ WSR-2 $W = 30233.5$, $e = 0.00$, $n1 = n2 = 12460$, $p = 0.00$
Plugin	Ms: $[3.0, 2.0]$ WSR-2 $W = 902595.0$, $e = 0.01$, $n1 = n2 = 12460$, $p = 0.00$
Screen	Ms: $[14.0, 13.0]$ WSR-2 $W = 1427159.5$, $e = 0.02$, $n1 = n2 = 12460$, $p = 0.00$
Storage	Ms: $[29.0, 24.0]$ WSR-2 $W = 3907641.5$, $e = 0.05$, $n1 = n2 = 12460$, $p = 0.00$
WebGL	Ms: $[0.0, 0.0]$ WSR-2 $W = 444.5$, $e = 0.00$, $n1 = n2 = 12460$, $p = 0.00$
WebGLext	Ms: $[0.0, 0.0]$ WSR-2 $W = 0.0$, $e = 0.00$, $n1 = n2 = 12460$, $p = 0.00$
WebRTC	Ms: $[0.0, 0.0]$ WSR-2 $W = 188.5$, $e = 0.00$, $n1 = n2 = 12460$, $p = 0.31$
Selenium vs non-Selenium driver	
Requests	

First-party	Ms: $[13.0, 13.0, 12.0, 16.0]$ $\chi^2(3) = 635.00$, $e = 0.11$, $p = 0.00$
FP-Ad-Track	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 20.52$, $e = 0.00$, $p = 0.00$
Third-Party	Ms: $[6.0, 5.0, 0.0, 8.0]$ $\chi^2(3) = 3689.07$, $e = 0.62$, $p = 0.00$
TP-Ad-Track	Ms: $[0.0, 0.0, 0.0, 2.0]$ $\chi^2(3) = 1814.77$, $e = 0.31$, $p = 0.00$
API Accesses	
Audio	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 0.83$, $e = 0.00$, $p = 0.84$
Automation	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 198.16$, $e = 0.03$, $p = 0.00$
Battery	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 298.80$, $e = 0.05$, $p = 0.00$
Canvas	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 28.09$, $e = 0.00$, $p = 0.00$
Configuration	Ms: $[3.0, 2.0, 2.0, 3.0]$ $\chi^2(3) = 66.16$, $e = 0.01$, $p = 0.00$
CSS	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 460.96$, $e = 0.08$, $p = 0.00$
Geolocation	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 15.68$, $e = 0.00$, $p = 0.01$
Plugin	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 890.80$, $e = 0.15$, $p = 0.00$
Screen	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 128.65$, $e = 0.02$, $p = 0.00$
Storage	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 81.06$, $e = 0.01$, $p = 0.00$
WebGL	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 174.27$, $e = 0.03$, $p = 0.00$
WebGLext	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 15.78$, $e = 0.00$, $p = 0.01$
WebRTC	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 5.64$, $e = 0.00$, $p = 0.26$
Static-scrolling comparison	
Requests	
First-party	Ms: $[34.0, 37.0, 35.0, 36.0]$ $\chi^2(3) = 1062.79$, $e = 0.03$, $p = 0.00$
FP-Ad-Track	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 95.93$, $e = 0.00$, $p = 0.00$
Third-Party	Ms: $[49.0, 51.0, 34.0, 35.0]$ $\chi^2(3) = 11681.81$, $e = 0.33$, $p = 0.00$
TP-Ad-Track	Ms: $[15.0, 16.0, 11.0, 12.0]$ $\chi^2(3) = 5173.84$, $e = 0.14$, $p = 0.00$
Dynamic-scrolling comparison	
Requests	
First-party	Ms: $[35.0, 44.0, 37.0, 40.0]$ $\chi^2(3) = 1238.16$, $e = 0.16$, $p = 0.00$
FP-Ad-Track	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 58.87$, $e = 0.01$, $p = 0.00$
Third-Party	Ms: $[80.0, 91.0, 59.0, 61.0]$ $\chi^2(3) = 2232.77$, $e = 0.29$, $p = 0.00$
TP-Ad-Track	Ms: $[40.0, 46.0, 30.0, 29.0]$ $\chi^2(3) = 1699.63$, $e = 0.22$, $p = 0.00$
Mainstream mobile vs desktop	
Requests	
First-party	Ms: $[32.0, 33.0, 32.0, 49.0]$ $\chi^2(3) = 13358.51$, $e = 0.31$, $p = 0.00$
FP-Ad-Track	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 4215.13$, $e = 0.10$, $p = 0.00$
Third-Party	Ms: $[48.0, 33.0, 30.0, 46.0]$ $\chi^2(3) = 17259.17$, $e = 0.40$, $p = 0.00$
TP-Ad-Track	Ms: $[14.0, 11.0, 14.0, 19.0]$ $\chi^2(3) = 5261.37$, $e = 0.12$, $p = 0.00$
Third-party tracking-and-advertising entities	
TP-Ad-Track	Ms: $[3.0, 3.0, 3.0, 3.0]$ $\chi^2(3) = 2087.28$, $e = 0.05$, $p = 0.00$
API Accesses	
Audio	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 49.59$, $e = 0.00$, $p = 0.00$
Automation	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 6257.52$, $e = 0.15$, $p = 0.00$
Battery	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 4648.81$, $e = 0.11$, $p = 0.00$
Canvas	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 730.45$, $e = 0.02$, $p = 0.00$
Configuration	Ms: $[28.0, 21.0, 26.0, 28.0]$ $\chi^2(3) = 3324.88$, $e = 0.08$, $p = 0.00$
CSS	Ms: $[36.0, 23.0, 29.0, 29.0]$ $\chi^2(3) = 2364.53$, $e = 0.06$, $p = 0.00$
Geolocation	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 156.80$, $e = 0.00$, $p = 0.00$
Plugin	Ms: $[2.0, 4.0, 2.0, 2.0]$ $\chi^2(3) = 7394.04$, $e = 0.17$, $p = 0.00$
Screen	Ms: $[11.0, 6.0, 10.0, 8.0]$ $\chi^2(3) = 4194.24$, $e = 0.10$, $p = 0.00$
Storage	Ms: $[19.5, 11.0, 17.0, 16.0]$ $\chi^2(3) = 3251.32$, $e = 0.08$, $p = 0.00$
WebGL	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 947.29$, $e = 0.02$, $p = 0.00$
WebGLext	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 628.67$, $e = 0.01$, $p = 0.00$
WebRTC	Ms: $[0.0, 0.0, 0.0, 0.0]$ $\chi^2(3) = 53.53$, $e = 0.00$, $p = 0.00$
Mobile privacy vs mainstream	
Requests	
First-party	Ms: $[31.0, 31.0]$ MW-2 $W = 207332798.5$, $e = 82933119.40$, $n1 = 32270, n2 = 12908$, $p = 0.45$
FP-Ad-Track	Ms: $[0.0, 0.0]$ MW-2 $W = 194454082.5$, $e = 77781633.00$, $n1 = 32270, n2 = 12908$, $p = 0.00$
Third-Party	Ms: $[23.0, 39.0]$ MW-2 $W = 144938698.0$, $e = 57975479.20$, $n1 = 32270, n2 = 12908$, $p = 0.00$

TP-Ad-Track	Ms: $[1.0, 13.0]$ MW-2 $W = 115292059.0$, $e = 46116823.60$, $n1 = 32270, n2 = 12908$, $p = 0.00$
Third-party tracking-and-advertising entities	
TP-Ad-Track	Ms: $[1.0, 3.0]$ MW-2 $W = 127995861.0$, $e = 51198344.40$, $n1 = 32270, n2 = 12908$, $p = 0.00$
API Accesses	
Audio	Ms: $[0.0, 0.0]$ MW-2 $W = 206309182.0$, $e = 82523672.80$, $n1 = 32270, n2 = 12908$, $p = 0.00$
Automation	Ms: $[0.0, 0.0]$ MW-2 $W = 208461834.5$, $e = 83384733.80$, $n1 = 32270, n2 = 12908$, $p = 0.75$
Battery	Ms: $[0.0, 0.0]$ MW-2 $W = 197821681.0$, $e = 79128672.40$, $n1 = 32270, n2 = 12908$, $p = 0.00$
Canvas	Ms: $[0.0, 0.0]$ MW-2 $W = 196218463.0$, $e = 78487385.20$, $n1 = 32270, n2 = 12908$, $p = 0.00$
Configuration	Ms: $[14.0, 26.0]$ MW-2 $W = 156905816.0$, $e = 62762326.40$, $n1 = 32270, n2 = 12908$, $p = 0.00$
CSS	Ms: $[22.0, 31.0]$ MW-2 $W = 197390698.5$, $e = 78956279.40$, $n1 = 32270, n2 = 12908$, $p = 0.00$
Geolocation	Ms: $[0.0, 0.0]$ MW-2 $W = 202853023.5$, $e = 81141209.40$, $n1 = 32270, n2 = 12908$, $p = 0.00$
Orientation	Ms: $[0.0, 0.0]$ MW-2 $W = 196581024.5$, $e = 78632409.80$, $n1 = 32270, n2 = 12908$, $p = 0.00$
Plugin	Ms: $[0.0, 2.0]$ MW-2 $W = 124349292.0$, $e = 49739716.80$, $n1 = 32270, n2 = 12908$, $p = 0.00$
Screen	Ms: $[2.0, 10.0]$ MW-2 $W = 128551457.0$, $e = 51420582.80$, $n1 = 32270, n2 = 12908$, $p = 0.00$
Storage	Ms: $[4.0, 17.0]$ MW-2 $W = 163406441.5$, $e = 65362576.60$, $n1 = 32270, n2 = 12908$, $p = 0.00$
WebGL	Ms: $[0.0, 0.0]$ MW-2 $W = 205280915.5$, $e = 82112366.20$, $n1 = 32270, n2 = 12908$, $p = 0.00$
WebGLext	Ms: $[0.0, 0.0]$ MW-2 $W = 199089793.5$, $e = 79635917.40$, $n1 = 32270, n2 = 12908$, $p = 0.00$
WebRTC	Ms: $[0.0, 0.0]$ MW-2 $W = 207208807.5$, $e = 82883523.00$, $n1 = 32270, n2 = 12908$, $p = 0.00$
Privacy-focused mobile vs desktop	
Requests	
First-party	Ms: $[31.0, 33.0]$ WSR-2 $W = 98981165.0$, $e = 0.23$, $n1 = n2 = 29318$, $p = 0.00$
FP-Ad-Track	Ms: $[0.0, 0.0]$ WSR-2 $W = 670602.5$, $e = 0.00$, $n1 = n2 = 29318$, $p = 0.00$
Third-Party	Ms: $[38.0, 38.0]$ WSR-2 $W = 197782498.0$, $e = 0.46$, $n1 = n2 = 29318$, $p = 0.00$
TP-Ad-Track	Ms: $[1.0, 1.0]$ WSR-2 $W = 22231393.5$, $e = 0.05$, $n1 = n2 = 29318$, $p = 0.00$
Third-party tracking-and-advertising entities	
TP-Ad-Track	Ms: $[1.0, 1.0]$ WSR-2 $W = 5636394.5$, $e = 0.01$, $n1 = n2 = 29318$, $p = 0.00$
API Accesses	
Audio	Ms: $[0.0, 0.0]$ WSR-2 $W = 1104.5$, $e = 0.00$, $n1 = n2 = 29318$, $p = 0.18$
Automation	Ms: $[0.0, 0.0]$ WSR-2 $W = 90153.0$, $e = 0.00$, $n1 = n2 = 29318$, $p = 0.00$
Battery	Ms: $[0.0, 0.0]$ WSR-2 $W = 323.5$, $e = 0.00$, $n1 = n2 = 29318$, $p = 0.00$
Canvas	Ms: $[0.0, 0.0]$ WSR-2 $W = 1725138.5$, $e = 0.00$, $n1 = n2 = 29318$, $p = 0.00$
Configuration	Ms: $[15.0, 12.0]$ WSR-2 $W = 41728423.0$, $e = 0.10$, $n1 = n2 = 29318$, $p = 0.00$
CSS	Ms: $[23.0, 13.0]$ WSR-2 $W = 47548281.0$, $e = 0.11$, $n1 = n2 = 29318$, $p = 0.00$
Geolocation	Ms: $[0.0, 0.0]$ WSR-2 $W = 1877.0$, $e = 0.00$, $n1 = n2 = 29318$, $p = 0.00$
Plugin	Ms: $[0.0, 0.0]$ WSR-2 $W = 8141242.5$, $e = 0.02$, $n1 = n2 = 29318$, $p = 0.00$
Screen	Ms: $[3.0, 1.0]$ WSR-2 $W = 7938657.5$, $e = 0.02$, $n1 = n2 = 29318$, $p = 0.00$
Storage	Ms: $[4.0, 2.0]$ WSR-2 $W = 16820777.0$, $e = 0.04$, $n1 = n2 = 29318$, $p = 0.00$
WebGL	Ms: $[0.0, 0.0]$ WSR-2 $W = 25536.0$, $e = 0.00$, $n1 = n2 = 29318$, $p = 0.00$
WebGLext	Ms: $[0.0, 0.0]$ WSR-2 $W = 63598.5$, $e = 0.00$, $n1 = n2 = 29318$, $p = 0.00$
WebRTC	Ms: $[0.0, 0.0]$ WSR-2 $W = 318.0$, $e = 0.00$, $n1 = n2 = 29318$, $p = 0.31$
Mobile privacy browser comparison	
Requests	
First-party	Ms: $[30.0, 35.0, 30.0, 30.0, 30.0, 31.0]$ $\chi^2(5) = 13266.76$, $e = 0.41$, $p = 0.00$
FP-Ad-Track	Ms: $[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]$ $\chi^2(5) = 4700.25$, $e = 0.14$, $p = 0.00$
Third-Party	Ms: $[41.0, 11.0, 13.0, 10.0, 29.0, 30.0]$ $\chi^2(5) = 17881.97$, $e = 0.55$, $p = 0.00$
TP-Ad-Track	Ms: $[0.0, 1.0, 2.0, 1.0, 13.0, 14.0]$ $\chi^2(5) = 20143.08$, $e = 0.62$, $p = 0.00$
API Accesses	
Audio	Ms: $[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]$ $\chi^2(5) = 267.46$, $e = 0.01$, $p = 0.00$
Automation	Ms: $[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]$ $\chi^2(5) = 3205.47$, $e = 0.10$, $p = 0.00$
Battery	Ms: $[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]$ $\chi^2(5) = 2093.22$, $e = 0.06$, $p = 0.00$

Canvas	Ms: $[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]$ $\chi^2(5) = 1977.78$, $e = 0.06$, $p = 0.00$
Configuration	Ms: $[10.0, 12.0, 15.0, 13.0, 24.0, 26.0]$ $\chi^2(5) = 12290.93$, $e = 0.38$, $p = 0.00$
CSS	Ms: $[19.0, 24.0, 22.0, 20.0, 26.0, 28.0]$ $\chi^2(5) = 2177.32$, $e = 0.07$, $p = 0.00$
Geolocation	Ms: $[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]$ $\chi^2(5) = 1020.50$, $e = 0.03$, $p = 0.00$
Orientation	Ms: $[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]$ $\chi^2(5) = 1392.75$, $e = 0.04$, $p = 0.00$
Plugin	Ms: $[0.0, 0.0, 0.0, 0.0, 2.0, 2.0]$ $\chi^2(5) = 14260.06$, $e = 0.44$, $p = 0.00$
Screen	Ms: $[0.0, 0.0, 2.0, 0.0, 9.0, 9.0]$ $\chi^2(5) = 15425.69$, $e = 0.48$, $p = 0.00$
Storage	Ms: $[1.0, 4.0, 6.0, 3.0, 13.0, 17.0]$ $\chi^2(5) = 7773.47$, $e = 0.24$, $p = 0.00$
WebGL	Ms: $[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]$ $\chi^2(5) = 909.98$, $e = 0.03$, $p = 0.00$
WebGLext	Ms: $[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]$ $\chi^2(5) = 562.14$, $e = 0.02$, $p = 0.00$
WebRTC	Ms: $[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]$ $\chi^2(5) = 122.43$, $e = 0.00$, $p = 0.00$
Desktop mainstream browser location comparison	
Requests	
First-party	Ms: $[39.0, 34.0]$ WSR-2 $W = 25857342.0$, $e = 0.12$, $n1 = n2 = 20488$, $p = 0.00$
FP-Ad-Track	Ms: $[0.0, 0.0]$ WSR-2 $W = 1262064.0$, $e = 0.01$, $n1 = n2 = 20488$, $p = 0.00$
Third-Party	Ms: $[37.0, 42.0]$ WSR-2 $W = 67049737.5$, $e = 0.32$, $n1 = n2 = 20488$, $p = 0.00$
TP-Ad-Track	Ms: $[14.0, 14.0]$ WSR-2 $W = 41677315.5$, $e = 0.20$, $n1 = n2 = 20488$, $p = 0.00$
API Accesses	
Audio	Ms: $[0.0, 0.0]$ WSR-2 $W = 11854.5$, $e = 0.00$, $n1 = n2 = 20488$, $p = 0.00$
Automation	Ms: $[0.0, 0.0]$ WSR-2 $W = 501491.0$, $e = 0.00$, $n1 = n2 = 20488$, $p = 0.63$
Battery	Ms: $[0.0, 0.0]$ WSR-2 $W = 38665.5$, $e = 0.00$, $n1 = n2 = 20488$, $p = 0.00$
Canvas	Ms: $[0.0, 0.0]$ WSR-2 $W = 3890345.0$, $e = 0.02$, $n1 = n2 = 20488$, $p = 0.00$
Configuration	Ms: $[24.0, 26.0]$ WSR-2 $W = 35138389.0$, $e = 0.17$, $n1 = n2 = 20488$, $p = 0.00$
CSS	Ms: $[24.0, 37.0]$ WSR-2 $W = 24751544.5$, $e = 0.12$, $n1 = n2 = 20488$, $p = 0.00$
Geolocation	Ms: $[0.0, 0.0]$ WSR-2 $W = 144654.0$, $e = 0.00$, $n1 = n2 = 20488$, $p = 0.24$
Plugin	Ms: $[3.0, 4.0]$ WSR-2 $W = 13011204.0$, $e = 0.06$, $n1 = n2 = 20488$, $p = 0.00$
Screen	Ms: $[7.0, 10.0]$ WSR-2 $W = 17160618.5$, $e = 0.08$, $n1 = n2 = 20488$, $p = 0.00$
Storage	Ms: $[12.0, 17.0]$ WSR-2 $W = 22470767.0$, $e = 0.11$, $n1 = n2 = 20488$, $p = 0.00$
WebGL	Ms: $[0.0, 0.0]$ WSR-2 $W = 50387.0$, $e = 0.00$, $n1 = n2 = 20488$, $p = 0.00$
WebGLext	Ms: $[0.0, 0.0]$ WSR-2 $W = 458317.5$, $e = 0.00$, $n1 = n2 = 20488$, $p = 0.05$
WebRTC	Ms: $[0.0, 0.0]$ WSR-2 $W = 3137.0$, $e = 0.00$, $n1 = n2 = 20488$, $p = 0.00$
Mobile mainstream browser location comparison	
Requests	
First-party	Ms: $[31.0, 31.0]$ WSR-2 $W = 22574578.0$, $e = 0.07$, $n1 = n2 = 25346$, $p = 0.89$
FP-Ad-Track	Ms: $[0.0, 0.0]$ WSR-2 $W = 564379.0$, $e = 0.00$, $n1 = n2 = 25346$, $p = 0.89$
Third-Party	Ms: $[38.0, 37.0]$ WSR-2 $W = 101819482.0$, $e = 0.32$, $n1 = n2 = 25346$, $p = 0.00$
TP-Ad-Track	Ms: $[13.0, 12.0]$ WSR-2 $W = 46696229.5$, $e = 0.15$, $n1 = n2 = 25346$, $p = 0.00$
API Accesses	
Audio	Ms: $[0.0, 0.0]$ WSR-2 $W = 4067.0$, $e = 0.00$, $n1 = n2 = 25346$, $p = 1.00$
Automation	Ms: $[0.0, 0.0]$ WSR-2 $W = 39511.0$, $e = 0.00$, $n1 = n2 = 25346$, $p = 0.00$
Battery	Ms: $[0.0, 0.0]$ WSR-2 $W = 55765.5$, $e = 0.00$, $n1 = n2 = 25346$, $p = 0.00$
Canvas	Ms: $[0.0, 0.0]$ WSR-2 $W = 1290565.5$, $e = 0.00$, $n1 = n2 = 25346$, $p = 0.29$
Configuration	Ms: $[25.0, 24.0]$ WSR-2 $W = 16413878.5$, $e = 0.05$, $n1 = n2 = 25346$, $p = 0.00$
CSS	Ms: $[28.0, 27.0]$ WSR-2 $W = 21845818.0$, $e = 0.07$, $n1 = n2 = 25346$, $p = 1.00$
Geolocation	Ms: $[0.0, 0.0]$ WSR-2 $W = 52993.5$, $e = 0.00$, $n1 = n2 = 25346$, $p = 0.00$
Orientation	Ms: $[0.0, 0.0]$ WSR-2 $W = 152488.5$, $e = 0.00$, $n1 = n2 = 25346$, $p = 0.00$
Plugin	Ms: $[2.0, 2.0]$ WSR-2 $W = 3322200.0$, $e = 0.01$, $n1 = n2 = 25346$, $p = 0.00$
Screen	Ms: $[9.0, 9.0]$ WSR-2 $W = 14390598.5$, $e = 0.04$, $n1 = n2 = 25346$, $p = 1.00$
Storage	Ms: $[17.0, 15.0]$ WSR-2 $W = 17027372.5$, $e = 0.05$, $n1 = n2 = 25346$, $p = 0.00$
WebGL	Ms: $[0.0, 0.0]$ WSR-2 $W = 9376.5$, $e = 0.00$, $n1 = n2 = 25346$, $p = 0.29$
WebGLext	Ms: $[0.0, 0.0]$ WSR-2 $W = 172081.0$, $e = 0.00$, $n1 = n2 = 25346$, $p = 0.00$
WebRTC	Ms: $[0.0, 0.0]$ WSR-2 $W = 1974.5$, $e = 0.00$, $n1 = n2 = 25346$, $p = 1.00$
Mobile privacy browser location comparison	
Requests	
First-party	Ms: $[32.0, 32.0]$ WSR-2 $W = 2063538.5$, $e = 0.07$, $n1 = n2 = 7865$, $p = 0.32$
FP-Ad-Track	Ms: $[0.0, 0.0]$ WSR-2 $W = 32917.0$, $e = 0.00$, $n1 = n2 = 7865$, $p = 1.00$
Third-Party	Ms: $[23.0, 23.0]$ WSR-2 $W = 7210762.0$, $e = 0.23$, $n1 = n2 = 7865$, $p = 1.00$
TP-Ad-Track	Ms: $[1.0, 1.0]$ WSR-2 $W = 1182248.5$, $e = 0.04$, $n1 = n2 = 7865$, $p = 0.00$

Desktop privacy browser location comparison	
Requests	
First-party	Ms: $[32.0, 33.0]$ WSR-2 $W = 17316676.0$, $e = 0.07$, $n1 = n2 = 23036$, $p = 0.00$
FP-Ad-Track	Ms: $[0.0, 0.0]$ WSR-2 $W = 172898.0$, $e = 0.00$, $n1 = n2 = 23036$, $p = 0.00$
Third-Party	Ms: $[35.0, 60.0]$ WSR-2 $W = 28078233.0$, $e = 0.11$, $n1 = n2 = 23036$, $p = 0.00$
TP-Ad-Track	Ms: $[1.0, 1.0]$ WSR-2 $W = 6201978.5$, $e = 0.02$, $n1 = n2 = 23036$, $p = 0.00$
Identical desktop Chrome	
Requests	
First-party	Ms: $[32.0, 33.0]$ WSR-2 $W = 6351146.5$, $e = 0.05$, $n1 = n2 = 15577$, $p = 0.36$
FP-Ad-Track	Ms: $[0.0, 0.0]$ WSR-2 $W = 146367.5$, $e = 0.00$, $n1 = n2 = 15577$, $p = 0.43$
Third-Party	Ms: $[34.0, 34.0]$ WSR-2 $W = 25662234.5$, $e = 0.21$, $n1 = n2 = 15577$, $p = 0.36$
TP-Ad-Track	Ms: $[11.0, 11.0]$ WSR-2 $W = 11391107.5$, $e = 0.09$, $n1 = n2 = 15577$, $p = 0.17$
Entities	
TP-Ad-Track	Ms: $[11.0, 11.0]$ MW-2 $W = 121095987.5$, $e = 121095987.50$, $n1 = 15577, n2 = 15577$, $p = 0.78$
API accesses	
Audio	Ms: $[0.0, 0.0]$ WSR-2 $W = 1268.5$, $e = 0.00$, $n1 = n2 = 15577$, $p = 1.00$
Battery	Ms: $[0.0, 0.0]$ WSR-2 $W = 70478.5$, $e = 0.00$, $n1 = n2 = 15577$, $p = 1.00$
Canvas	Ms: $[0.0, 0.0]$ WSR-2 $W = 554852.5$, $e = 0.00$, $n1 = n2 = 15577$, $p = 1.00$
Configuration	Ms: $[21.0, 21.0]$ WSR-2 $W = 8788251.5$, $e = 0.07$, $n1 = n2 = 15577$, $p = 0.62$
CSS	Ms: $[23.0, 23.0]$ WSR-2 $W = 5972573.0$, $e = 0.05$, $n1 = n2 = 15577$, $p = 0.76$
Geolocation	Ms: $[0.0, 0.0]$ WSR-2 $W = 12231.0$, $e = 0.00$, $n1 = n2 = 15577$, $p = 1.00$
Plugin	Ms: $[4.0, 4.0]$ WSR-2 $W = 1939014.5$, $e = 0.02$, $n1 = n2 = 15577$, $p = 0.27$
Screen	Ms: $[6.0, 6.0]$ WSR-2 $W = 4416821.5$, $e = 0.04$, $n1 = n2 = 15577$, $p = 0.27$
Storage	Ms: $[11.0, 11.0]$ WSR-2 $W = 6820344.0$, $e = 0.06$, $n1 = n2 = 15577$, $p = 0.19$
WebGL	Ms: $[0.0, 0.0]$ WSR-2 $W = 2286.0$, $e = 0.00$, $n1 = n2 = 15577$, $p = 1.00$
WebGLext	Ms: $[0.0, 0.0]$ WSR-2 $W = 110072.0$, $e = 0.00$, $n1 = n2 = 15577$, $p = 0.12$
WebRTC	Ms: $[0.0, 0.0]$ WSR-2 $W = 430.0$, $e = 0.00$, $n1 = n2 = 15577$, $p = 1.00$