Zhiju Yang and Chuan Yue*

A Comparative Measurement Study of Web Tracking on Mobile and Desktop Environments

Abstract: Web measurement is a powerful approach to studying various tracking practices that may compromise the privacy of millions of users. Researchers have built several measurement frameworks and performed a few studies to measure web tracking on the desktop environment. However, little is known about web tracking on the mobile environment, and no tool is readily available for performing a comparative measurement study on mobile and desktop environments. In this work, we built a framework called WTPatrol that allows us and other researchers to perform web tracking measurement on both mobile and desktop environments. Using WTPatrol, we performed the first comparative measurement study of web tracking on 23,310 websites that have both mobile version and desktop version webpages. We conducted an in-depth comparison of the web tracking practices of those websites between mobile and desktop environments from two perspectives: web tracking based on JavaScript APIs and web tracking based on HTTP cookies. Overall, we found that mobile web tracking has its unique characteristics especially due to mobile-specific trackers, and it has become increasingly as prevalent as desktop web tracking. However, the potential impact of mobile web tracking is more severe than that of desktop web tracking because a user may use a mobile device frequently in different places and be continuously tracked. We further gave some suggestions to web users, developers, and researchers to defend against web tracking.

Keywords: Web tracking, measurement, user privacy, mobile, JavaScript API

DOI 10.2478/popets-2020-0016
Received 2019-08-31; revised 2019-12-15; accepted 2019-12-16.

Zhiju Yang: Colorado School of Mines, E-mail: zhijuyang@mines.edu
***Corresponding Author: Chuan Yue:** Colorado School of Mines, E-mail: chuanyue@mines.edu

1 Introduction

Web tracking is frequently performed over the Internet by various trackers to collect the information of users' browsing activities for various purposes including personalized advertisement, targeted attacks, and surveillance [12, 21, 32]. Traditionally, stateful HTTP cookies are used as the dominant technique to track online users [27]. In recent years, advanced stateful tracking techniques such as Flash cookies and advanced stateless tracking techniques such as browser or device fingerprinting have also become very popular over the Internet [1, 3, 24, 30]. User privacy has been keeping compromised by trackers that utilize these techniques.

Web measurement is a powerful approach to studying online tracking practices and techniques. Researchers have built several measurement frameworks and performed a few studies to measure web tracking on the desktop environment [2, 11, 19, 21]. However, almost all the existing measurement studies were performed on the desktop environment. Little is known about web tracking on the mobile environment, and no tool is readily available for performing a comparative measurement study on mobile and desktop environments. To the best of our knowledge, only Eubank et al. [13] measured the role played by HTTP cookies and JavaScript in mobile web tracking on 500 websites.

Measuring web tracking specifically on the mobile environment is necessary and important because of the following reasons. First, mobile web browsing has been booming and overtaking the desktop web browsing over the years [35]. Second, web tracking measurement on the desktop environment cannot represent the complete web tracking practices because many websites have mobile-specific versions which could have different tracking practices. Third, certain web techniques such as plugin support are implemented differently between mobile and desktop browsers, which could lead to web tracking practice differences. Fourth, many web techniques and events such as those device sensor related ones are mobile-specific, and can also be leveraged for tracking mobile web users. Therefore, measuring mobile web tracking by simply using a desktop browser to

mimic a mobile browser may lead to inaccurate results as we verified in an experiment (Section 3).

In this work, we built a framework called WTPatrol that allows us and other researchers to perform web tracking measurement on both mobile and desktop environments. WTPatrol consists of three modules: *automation driver* which serves to automatically control the crawling behavior of a browser, *WTPatrol browser extension* which instruments a browser for detecting a variety of web tracking practices, and *data analyzer* which analyzes and compares web tracking related data collected from mobile and desktop websites.

Using WTPatrol, we performed the first comparative measurement study of web tracking on 23,310 websites that have both mobile version and desktop version webpages. We conducted an in-depth comparison of web tracking practices of those websites between mobile and desktop environments from two perspectives: tracking based on JavaScript APIs and based on HTTP cookies.

From the perspective of JavaScript APIs, we in total identified 5,835 different trackers [1] and found that (1) 762 (13.1%) trackers are mobile-specific because they only appeared on mobile websites, (2)1,783 (30.6%) trackers are desktop-specific because they only appeared on desktop websites, and (3) 3,290 (56.3%) trackers appeared on both mobile and desktop websites. From the perspective of HTTP cookies, we in total identified 5,574 different trackers and found that (1) 695 (12.5%) trackers are mobile-specific because they only placed cookies on mobile websites, (2) 1,536 (27.6%) trackers are desktop-specific because they only placed cookies on desktop websites, and (3) 3,343 (59.9%) trackers placed cookies on both mobile and desktop websites. Overall, we found that mobile web tracking has its unique characteristics especially due to mobile-specific trackers, and it has become increasingly as prevalent as desktop web tracking in terms of the overall volume of trackers. However, the potential impact of mobile web tracking is more severe than that of desktop web tracking because a user may use a mobile device frequently in different places and be continuously tracked.

Our paper makes the following major contributions: (1) we built a web tracking measurement framework, WTPatrol, that allows researchers to perform web tracking measurement studies on both mobile and desktop environments; (2) we performed the first comparative measurement study of web tracking on 23,310 websites that have both mobile version and desktop version webpages; (3) we conducted an in-depth comparison of web tracking practices of those websites between mobile and desktop environments from the perspectives of JavaScript APIs and HTTP cookies; (4) we gave some suggestions to web users, developers, and researchers to defend against web tracking.

Section 2 reviews the related work. Section 3 describes the design of our WTPatrol measurement framework. Section 4 presents our data collection, dataset, and tracker definition. Section 5 presents and analyzes our measurement results. Section 6 gives our suggestions. Section 7 concludes this paper.

2 Related Work

Web tracking compromises the privacy of users by associating their identities with their browsing activities on different websites. In this section, we review the studies on investigating the effectiveness of web tracking techniques, and the studies on measuring the adoption of different web tracking techniques in the wild.

2.1 Effectiveness of Web Tracking Techniques

In general, there are two types of web tracking techniques: stateful and stateless. In term of stateful techniques, a tracking website usually generates some unique identifiers, saves them to users' machines, and tracks users when the identifiers are carried back. In contrast, stateless techniques are usually fingerprinting-based, and a tracking website derives unique user identifiers or fingerprints from the attributes of browsers, operating systems, and/or devices.

HTTP cookies are the oldest stateful tracking techniques [17], but recent studies such as the one performed by Roesner et al. [27] showed that they are still widely used on many websites. In addition to HTTP cookies, many advanced stateful web tracking techniques have also widely appeared over the years. For example, supercookies are Adobe Flash files, HTTP Etag is an HTTP response header field, and HTML5 local storage is a browser's local storage for large data objects; they can all be used to store stateful information at the client-side and track web users [1, 3, 24, 30].

Due to the awareness of privacy issues caused by stateful techniques and the capability for privacy-

1 A tracker in this paper is identified at the granularity of the fully qualified domain name (FQDN) [50].

conscious users to disable such techniques, more websites have started to use stateless web tracking techniques such as browser fingerprinting. Browser fingerprinting identifies unique web browsers based on their attributes. Researchers often evaluate the effectiveness of browser fingerprinting techniques by setting up a website to invite visitors and collect the attributes of their browsers. For example, in the influential Panopticlick study [10], Eckersley collected 470,161 browser fingerprints composed of 10 attributes, and found that 83.6% of those browser fingerprints are unique. Similar to the Panopticlick study, Laperdrix et al. collected 118,934 browser fingerprints composed of 17 attributes [18], and found that 90% of desktop browser fingerprints and 81% of mobile browser fingerprints are unique, respectively.

Researchers also demonstrated the fingerprintability of some other individual attributes such as those of browser extensions [31], JavaScript engines [22, 23], and HTML5 battery status [25]. Meanwhile, some researchers showed the fingerprintability of smartphones by exploiting the manufacturing imperfections in hardware devices such as speakers and microphones [4, 7, 34] as well as accelerometer and gyroscope motion sensors [4, 8, 9]. It is worth noting that third-party tracking also occurs in mobile apps [14, 15, 20], although the focus of our work is on browser-based tracking.

2.2 Web Tracking Measurement Studies

To explore the extent to which web tracking techniques reviewed above are adopted in the wild, researchers have developed different tools and conducted several measurement studies.

Mayer et al. [21] implemented a Firefox browser extension, *FourthParty*, which can intercept HTTP traffic and monitor Document Object Model (DOM) events. They investigated third-party web tracking policies and techniques by using this tool over 500 websites. Acar et al. [2] developed a measurement tool, *FPDetective*, which is built upon a stripped-down browser PhantomJS and the Chromium browser driven by Selenium [56]. They measured the top million Alexa websites, identified new fingerprinting scripts and Flash objects, and found that web tracking techniques are widely used. More recently, Englehardt et al. [11] developed *OpenWPM*, which is a comprehensive web privacy measurement platform that directly leverages a full-fledged Firefox browser. They also measured the top million Alexa websites, and showed the prevalence of many web tracking techniques. Das et al. [6] developed OpenWPM-Mobile which still runs on a desktop environment; they found a significant overlap between tracking scripts and scripts accessing mobile sensor data. Lerner et al. [19] analyzed the Internet archive data from 1996 to 2016, and found that web tracking has become more sophisticated and pervasive over the years.

Existing measurement studies exemplified above were all performed using desktop computers. Measuring mobile web tracking directly using mobile devices is necessary and important (Section 1), and it needs the support from tools that ideally leverage a full-fledged mobile browser. To the best of our knowledge, only Eubank et al. measured mobile web tracking using a mobile browser [13]. In more details, they ported the *FourthParty* extension to the mobile Firefox, and measured the role of HTTP cookies and JavaScript played in mobile web tracking on 500 websites. In contrast, we performed the first comparative measurement study on 23,310 websites, and conducted an in-depth analysis of JavaScript API based and HTTP cookie based web tracking practices between mobile and desktop environments.

3 Measurement Framework

A few measurement tools are available on desktop environments as described in Section 2.2. However, there is no measurement framework readily available for researchers to perform comparative web privacy measurement studies on mobile and desktop environments, despite that mobile web browsing has been booming and overtaking the desktop web browsing over the years [35].

Measuring mobile web tracking directly using mobile devices (instead of using desktop computers without or with emulated mobile browser user agents) is necessary and important. This is because many differences between mobile and desktop websites can incur inaccuracy if we use a desktop-based tool to measure mobile websites. We verified this inaccuracy by examining the differences between visiting websites using a real mobile Firefox browser and using an emulated mobile Firefox browser (which is a desktop Firefox browser configured with the user agent string of a real mobile Firefox and the screen resolution of a smartphone).

In more details, we visited the homepages of the top 100 Alexa websites and compared the corresponding webpage sources between the real mobile browser and the emulated one. We found that 15 out of the top 100 Alexa websites returned different webpages (in terms of the DOM structure, JavaScript content, or

page content) to the two browsers. The proportional distances (Defined in Section 4.2) of returned webpages of these 15 websites vary from 0.03 to 0.95, with the average, standard deviation, and median values at 0.41, 0.34, and 0.27, respectively. Specifically, five websites returned two different versions of webpages to these two browsers (i.e., the emulated mobile browser was detected as a desktop browser, and the desktop version webpages are at least 38% larger in file size than the corresponding mobile version webpages). For the rest ten websites, eight of them returned webpages differing in at least one *script* element as exemplified in Table 5 of Appendix B, while two of them returned webpages differing in at least one large *div* element with more than six child elements. This result indicates that websites may use other techniques beyond checking the user agent string and screen resolution to detect the type of a browser, and would not return mobile content to an emulated mobile browser.

Note that Das et al. [6] built OpenWPM-Mobile to more realistically emulate Firefox for Android running on a real Moto G5 Plus smartphone. Because emulating devices is complicated as it requires manual fine-tuning for every new device that comes out, we in this work take the simpler approach of using real mobile devices although it has its own limitations. It is not very scalable in terms of using multiple (or multiple types of) devices and is relatively expensive, in comparison to the use of emulated mobile devices.

Therefore, we set out to develop a framework, WTPatrol, that can allow researchers to accurately measure web tracking practices on both mobile and desktop environments, and comprehensively perform comparative web privacy measurement studies. Figure 1 illustrates the architecture of WTPatrol with three modules: *automation driver* which automatically controls the crawling behavior of a browser, *WTPatrol browser extension* which instruments a browser for detecting a variety of web tracking practices, and *data analyzer* which analyzes and compares web tracking related data collected from mobile and desktop websites.

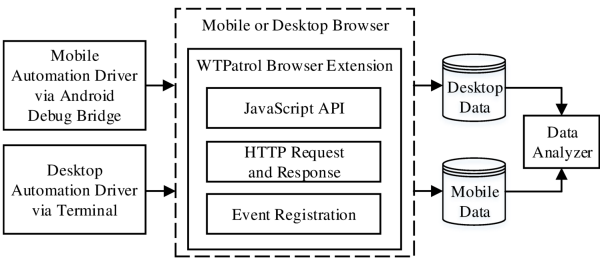


Fig. 1. The architecture of the WTPatrol framework.

3.1 WTPatrol Browser Extension

As reviewed in Section 2.2, OpenWPM [11] is a recent platform for measuring web tracking on desktop environments. It mainly uses a Firefox browser extension expanded from *FourthParty* [21] to measure web tracking practices. This approach of using a browser extension and a full-fledged browser enables more complete web tracking measurement than using a stripped-down browser such as PhantomJS [11]. Therefore, we build our WTPatrol browser extension upon the Firefox extension used in OpenWPM by porting it to mobile Firefox and further expanding it with new capabilities. Since a desktop Firefox extension does not automatically work on mobile Firefox, to begin with, we modified the code of the OpenWPM Firefox extension to make it compatible with mobile Firefox (specifically Firefox for Android) using the Firefox Add-on SDK. We then expanded this mobile version extension from three aspects.

First, we made the architectural change. OpenWPM integrates its data storage into a separate module written in Python instead of into its extension. While this approach lessens the burden of the Firefox browser, it increases the complexity and incurs new dependency by having the extra communication with the data storage module installed on the running platform. Therefore, to simplify the usage of our extension and remove the dependency on an extra data storage module, we integrated the data storage directly into the extension by taking the same approach as in *FourthParty* [21].

Second, we enriched the list of instrumentation for tracking techniques. As shown in Figure 1, our WTPatrol browser extension mainly measures three types of techniques that can be potentially utilized to track web users: JavaScript API, HTTP request and response, and event registration. Table 1 summarizes all the 34 JavaScript APIs (or objects) instrumented in the WTPatrol browser extension. We added new instruments to measure all the event handler registration activities with the initial goal to capture mobile-specific web tracking techniques. In more details, by instrumenting *window.addEventListener*, our WTPatrol browser extension is capable of, for example, capturing *devicemotion* events which are related to the motion sensor access, and *devicelight* events which are related to the ambient light sensor access.

Third, we added a self-crawling mode to the extension, so that it can automatically visit a list of websites one by one as OpenWPM does. For each website, our extension can close the previous browser tab and open a

new one to visit its homepage; it stays for a configured period of time before visiting the next website.

Based on this completed WTPatrol browser extension for mobile Firefox, we also created a desktop version with the same capabilities. So now web tracking measurement studies can be fairly performed and compared on both mobile and desktop environments.

Table 1. Thirty four JavaScript APIs (or objects) that can be utilized to perform web tracking.

Tracking Technique	Tracking Technique
window.navigator.userAgent	window.document.cookie
window.addEventListener	window.navigator.plugins
window.navigator.language	window.screen.colorDepth
window.localStorage	window.Storage
CanvasRenderingContext2D	window.name
window.navigator.platform	window.navigator.appName
window.sessionStorage	AudioContent
window.navigator.appVersion	window.navigator.product
window.navigator.vendor	HTMLCanvasElement
window.navigator.doNotTrack	window.performance
window.navigator.mimeTypes	window.screen.pixelDepth
window.navigator.onLine	window.navigator.oscpu
window.screen.orientation	ScriptProcessorNode
window.navigator.geolocation	RTCPeerConnection
window.navigator.buildID	AnalyserNode
window.navigator.cookieEnabled	OscillatorNode
window.navigator.appCodeName	GainNode

3.2 WTPatrol Automation Driver

Automation is essential for performing large-scale web measurement studies. For the sake of simplicity, we did not choose automation tools such as Selenium [56] to drive the Firefox browser and our browser extension; instead, we wrote shell scripts and leveraged Android Debug Bridge (ADB) to build our mobile version automation driver. Specifically, our shell scripts can programmatically start/stop the browser, prepare browsing profiles, and configure our WTPatrol browser extension. The shell scripts are executed on a desktop computer, and they communicate with an Android device via USB using the ADB service. In contrast, our desktop version automation driver executes shell scripts directly on a terminal to drive a desktop Firefox browser and our browser extension.

4 Data Collection, Dataset, and Tracker Definition

We perform measurements using both mobile devices and desktop computers to capture web tracking practice differences between the two environments. We now present data collection, dataset, and tracker definition.

4.1 Data Collection

In our study, we used four mobile devices, which are Google Nexus 6p smartphones installed with Firefox for Android and mobile WTPatrol. We also used four desktop computers installed with Ubuntu 14.04, desktop Firefox, and desktop WTPatrol. The version of both Firefox for Android and desktop Firefox is 53.0 without the tracking protection function. Each data collection experiment includes two stateless[2] measurements on the mobile and desktop environments, respectively. From our campus network, we ran the two stateless measurements side by side on the homepages of 116,000 websites[3], among which 100,000 websites are the top Alexa websites[4] and 16,000 websites are sampled lower rank websites. The way we sampled the lower rank websites is selecting the first 2,000 of every 100,000 websites within the rank from 200,000 to 1,000,000 based on the Alexa top one million site list, resulting in eight groups. After each homepage is loaded, WTPatrol collects the data for 20 seconds, among which the last five seconds are used to scroll down the current webpage five times with an interval of one second and a *scrollHeight* of the HTML *body* element. The rendered webpage source will be saved after the last scroll for analysis.

Considering the dynamic nature of web content (e.g., dynamic advertising and time differences), we performed two data collection experiments with one day difference between them using two sets of mobile devices and desktop computers. These two experiments, referred to as *Experiment A* and *Experiment B*, were performed from June 10th, 2019 to June 30th, 2019. That is, in both experiments, we collected data from all the 116,000 websites, but the data for each website in Experiment B was collected one day after that for the same website in Experiment A. Due to reasons such as timeout and site server errors, in total we have 96,093 and 98,335 successful website visits with webpage sources saved in Experiments A and B, respectively, and have 89,828 successful website visits in both experiments. We further identified from these 89,828 sites

2 With the browser state being cleaned for each website visit.
3 To speed up the data collection, we divided the 116,000 sites into roughly two halves, and used two sets of smartphones and desktop computers to crawl each half independently.
4 The Alexa top one million site list is dated on June 3rd, 2019.

a list of 23,310 sites that have both desktop and mobile versions of homepages. In other words, when each of these 23,310 sites is visited by a desktop browser and a mobile browser, respectively, the two returned homepages are not very similar, indicating that the website generates different webpages between the two environments and a comparison between them is meaningful.

4.2 A Dataset of 23,310 Sites

We are not aware of an existing list of websites that have both desktop and mobile versions, so we compare the pair of desktop and mobile webpages in terms of their similarity for each of those 89,828 websites to eventually identify that list of 23,310 sites as the dataset. There are many ways to compare the similarity between two webpages, ranging from simple but strict comparison on whether two pages are identical, to complex but flexible DOM tree edit distance calculation. We leveraged a simple yet flexible *proportional distance* metric proposed in [5] to quantify the similarity between two webpages. This metric is the ratio of the HTML tag-vector level Hamming Distance to the number of tags that appear in at least one of the two webpages, so that the former is normalized to take into account the webpage complexity differences among many different websites.

Formally, the proportional distance between two webpages A_1 and A_2 is calculated as follows [5]: (1) define an arbitrary but fixed ordering of a corpus of n HTML tags (n=107 as in [5], that is, we consider 107 different HTML tags which are from the complete set of tags provided by World Wide Web Consortium [60] after removing some of more common tags such as <head>, <body>, and <html> but including tags such as <script>), and define a "vector" as an array of n integers; (2) construct the vector t_1 for the webpage A_1 by counting the number of times each HTML tag of the corpus appears in A_1, and construct the corresponding vector t_2 for the webpage A_2; (3) for each pair of corresponding integers x_1 and x_2 in the two vectors t_1 and t_2, define the Hamming Distance $D(x_1, x_2) = 1$ if $x_1 \neq x_2$, and $D(x_1, x_2) = 0$ otherwise; (4) for each pair of corresponding integers x_1 and x_2 in the two vectors t_1 and t_2, further define $L(x_1, x_2) = 1$ if $x_1 \neq 0$ OR $x_2 \neq 0$, and $L(x_1, x_2) = 0$ otherwise, to indicate whether the corresponding tag appears in at least one of the two pages; (5) for non-null vectors t_1 and t_2, calculate the proportional distance $PD(t_1, t_2)$ between the two webpages A_1 and A_2 using Formula 1:

$$PD(t_1, t_2) = \frac{\sum_{i=1}^{n} D(t_1[i], t_2[i])}{\sum_{i=1}^{n} L(t_1[i], t_2[i])} \quad (1)$$

The authors in [5] used this metric to compare the similarity between two phishing webpages. We use this metric to compare the similarity between the mobile version and the desktop version webpages of the same site. Note that we provided an example of the proportional distance calculation in Appendix A.

Basically, the larger a proportional distance is, the bigger the difference between a pair of mobile webpage and desktop webpage, indicating that the given website is more likely to have two different versions on mobile and desktop environments. We calculated the proportional distance values of the 89,828 websites in Experiments A and B individually. We found that for 35.4% and 36.2% of all the 89,828 websites in Experiments A and B, respectively, the proportional distance between each pair of mobile webpage and desktop webpage is 0, indicating that the two versions are identical or nearly identical between mobile and desktop environments.

Strictly speaking, once the proportional distance is greater than 0, some differences exist between a mobile webpage and the corresponding desktop webpage. However, considering that some subtle differences could be incurred due to the dynamics of webpages such as time-sensitive strings and dropdown lists, choosing a threshold value that is greater than 0 but not too large will be helpful for us to suppress such noises.

To select a reasonable proportional distance value as the threshold for identifying whether a given website has two specific versions, we randomly sampled 700 websites from seven groups[5] that have different proportional distance values and manually inspected their webpage content. In more details, we started from the group of websites with the proportional distance value at 0.05, and found that their two versions of webpages do not differ significantly especially in terms of the JavaScript code and text content; we continued the manual inspection for the groups of websites with the proportional distance value at 0.1, 0.15,..., 0.35, respectively, and found that 0.35 is a reasonable threshold value for which at least 65% of the websites in the group differ significantly. There is always a trade-off between choosing a smaller and a larger threshold value in similarity comparison. In our case, choosing a smaller threshold value would include into our dataset more websites that do not have

5 With 100 websites from each group.

significant differences between their two versions of webpages, while choosing a larger threshold value would exclude from our dataset more websites that do have obvious differences between their two versions of webpages.

Therefore, we decided to use 0.35 as the threshold value, which is indeed close to the value 0.32 used in [5] for comparing phishing webpages. In total, we identified a list of 23,310 (25.9% of 89,828) sites with the proportional distance values greater than 0.35 in both Experiments A and B as the dataset for us to further compare their privacy practices. Figure 2 depicts the distribution of the 23,310 sites in six Alexa ranking groups. Note that the *selected* group is for sites with the rank between 200,000 and 1,000,000 as described at the beginning of this section. We observed a clear trend that the higher the ranking group, the larger the percentage of websites being selected from the group. This trend intuitively indicates that higher ranked websites are more likely to have both desktop-specific and mobile-specific versions of webpages than lower ranked ones.

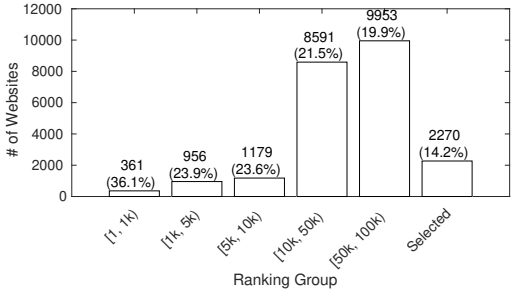


Fig. 2. Distribution of the 23,310 sites in six ranking groups.

4.3 Tracker Definition

A tracker in this paper is identified at the granularity of the fully qualified domain name (FQDN) [50]. We identify a tracker only if it appears in a third-party URL, i.e., it must be a third-party domain. To determine if a URL is a first-party URL or third-party URL, we leverage Mozilla's Public Suffix List [53], in which a public suffix is "one under which Internet users can (or historically could) directly register names." More specifically, we classify a URL as ***first-party*** if its public suffix together with the first section of the domain name proceeding the public suffix is identical to that of the URL displayed in the address bar of a browser; otherwise, we classify the URL as ***third-party***. This third-party classification method is commonly used in many studies such as [11].

We then classify a third-party domain as a ***tracker*** if it would be blocked by some ad block engine. In this paper, we leverage the ad block engine used in the Brave browser [39] to classify third-party domains as trackers or non-trackers. In more details, the engine first parses filters based on blacklists such as EasyList, and then matches third-party URLs against the parsed filters (e.g., regular expressions, URL substrings, and whitelisted rules). The blacklists used in our work are EasyPrivacy [49], EasyList [47], and 12 EasyList variants for different countries or languages [48]. Note that this collection of trackers may target at either desktop or mobile users as we inspected. While blacklists would never be complete to include all trackers, they are appropriate for us to capture the differences of web tracking on the two environments. After parsing and matching against these lists, a third-party domain is classified as a tracker if it would be blocked by the ad block engine. For instance, the first rule of the current EasyList is "&act=ads_"; example.com will be labeled as a tracker in the "example.com/index.html?x=1&act=ads_" context, but not in the "example.com/beningpage.html" context as no matching pattern is identified by the adblock engine. In other words, similar to [11], we classify a third-party domain as a tracker only if it is in the tracking context. Also note that well-known third-party libraries such as *jQuery* and *React* are usually not blocked by blacklists, and thus they are not considered as trackers in our study.

A tracker may perform tracking activities using JavaScript APIs or placing HTTP cookies. In this paper, we refer to *JavaScript API using trackers* and *cookie placement trackers* as ***JS-trackers*** and ***cookie-trackers***, respectively. Our WTPatrol browser extension records the JavaScript execution (Table 1) and cookie setting (e.g., via HTTP responses) activities associated with a visited first-party websites; therefore, in our measurements, we ensure that a tracker has actually used some of those techniques in practice, i.e., a tracker on a first-party webpage has executed JavaScript code or placed cookies that can be used for tracking.

In the rest of this paper, our analysis is based on the data collected in both Experiments A and B. We count a tracker as being included by a first-party website as long as certain tracking activities are identified in either Experiment A or Experiment B. Correspondingly, unique JavaScript APIs and cookies are counted as long as they are identified in either Experiment A or Experiment B.

5 Measurement Results Analysis

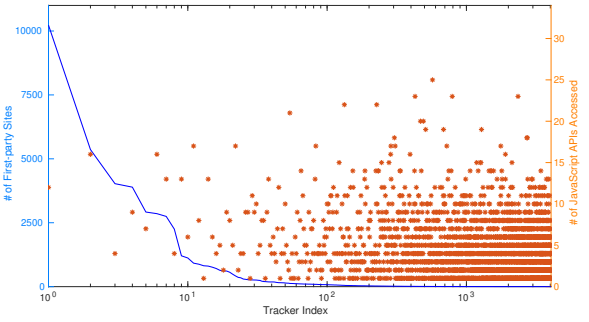
In this section, we present our measurement results and analysis from the perspective of tracking based on JavaScript APIs and the perspective of tracking based on HTTP cookies. We conclude this section with a summary of results and implications.

5.1 Tracking Based on JavaScript APIs

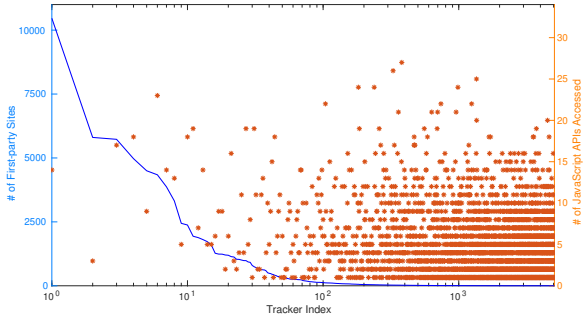
5.1.1 Trackers' Distribution and Their Techniques

Figure 3 presents the ***general view of all the JS-trackers***, i.e., 4,052 on the mobile environment and 5,073 on the desktop environment, in terms of the number of first-party sites they appeared on and the number of tracking JavaScript APIs they accessed. The y-axis on the left side and the blue curve represent the number of first-party websites that include each of all the trackers shown on the x-axis in log scale, while the y-axis on the right side and the red markers represent the number of JavaScript APIs (i.e., out of those 34 JavaScript APIs shown in Table 1) accessed by each of those trackers. The blue curves in Figure 3 show the "long-tail" phenomenon with trackers in the tail only appearing on very few websites. In more details, we found the vast majority of the trackers, i.e., 3,601 (88.9%) out of 4,052 trackers on the mobile environment and 4,389 (86.5%) out of 5,073 trackers on the desktop environment, only appear on less than ten first-party websites. In addition, the number of JavaScript APIs accessed by all the trackers vary from 1 to 27. On average, the trackers appear on 18 mobile websites and 24 desktop websites.

To estimate the false positives in our tracker identification, we manually analyzed 100 randomly sampled websites. False positives may occur in two situations. One is that a tracker is indeed not a third-party because it belongs to the same organization as its corresponding first-party, thus a false positive occurs. For example, *fbcdn.net* may be misclassified as a tracker on *facebook.com* while they belong to the same organization Facebook. The other situation is that a tracker is just a within-site (instead of cross-site) tracker according to Roesner et al. [27], thus a false positive occurs. A within-site tracker (e.g., *google-analytics.com* on a first-party website) cannot use unique identifiers to track users across sites while a cross-site tracker (e.g., *doubleclick.net* on a first-party website) can [27]. We randomly sampled 100 *(first-party domain, tracker domain)* pairs (with 100 different first-party domains and 100 different tracker domains) to estimate the false positives in our study. In terms of the first situation, we found that zero pair has the same organization. In terms of the second situation, by using the same two properties of a within-site tracker [6] as defined in [27], we identified six within-site trackers in the 100 sampled pairs. Overall, the false positive rate based on the sampled pairs is 6% considering both situations.



(a) Trackers on the mobile version first-party websites



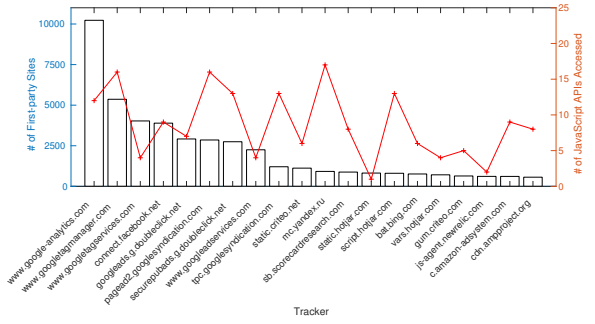
(b) Trackers on the desktop version first-party websites

Fig. 3. All the JS-trackers that appeared on the most number of those 23,310 first-party websites.

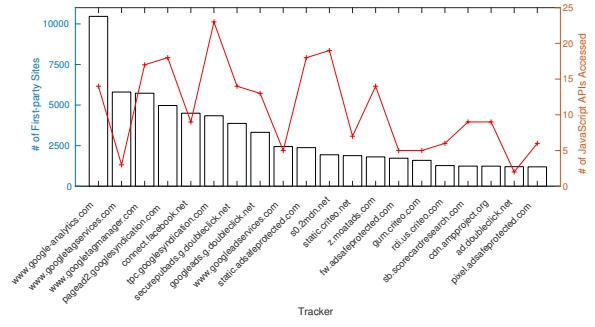
Figures 4a and 4b show the ***top 20 JS-trackers*** that appeared on the most number of those 23,310 first-party mobile version and desktop version websites, respectively. The meanings of the y-axes and x-axis are identical to those in Figure 3, except that the x-axis represents the top 20 trackers with their domain names, and the bars represent the number of first-party websites that include each of these 20 trackers. We found that *www.google-analytics.com* is the top tracker on both mo-

6 Property 1: the tracker's script, running in the context of the site, sets a site-owned cookie. Property 2: the tracker's script explicitly leaks the site-owned cookie in the parameters of a request to the tracker's domain, circumventing the same-origin policy.

bile and desktop websites, appearing on 10,464 (44.9%) and 10,228 (43.9%) of them, respectively. Note that, eight of the top ten trackers belong to the same organization, Google, indicating its dominant tracking power. Thirteen common trackers appear on both Figures 4a and 4b, which means that those top trackers monopolize web tracking on both mobile and desktop environments. In Figure 4, on average the top 20 trackers appear on 2,196 mobile websites accessing 9 JavaScript APIs and on 3,146 desktop websites accessing 11 JavaScript APIs.



(a) Trackers on the mobile version first-party websites



(b) Trackers on the desktop version first-party websites

Fig. 4. Top 20 JS-trackers that appeared on the most number of those 23,310 first-party websites.

We investigated ***20 randomly sampled trackers in each long tail***, and found that the average number of tracking JavaScript APIs accessed by these trackers is four on both environments (which is much less than that for the top 20 trackers). We further checked the trustworthiness of these 40 sampled trackers using a popular web reputation lookup service WebOfTrust (WOT) [58]. From 21 of them that have the reputation data, we found that the average reputation score is 65 and 58 on mobile and desktop environments, respectively. In comparison, we found that 20 trackers in Figure 4a and 19 trackers in Figure 4b have the reputation data with the average scores 66 and 60, respectively. Though the average reputation scores are close to each other between those sampled trackers and the top trackers, we found that the percentages of trackers that have the reputation grade lower than *Good* (i.e., the reputation score is less than 60) are different with 47.6% and 30.7% in the two groups, respectively.

We inspected ***all the JavaScript APIs accessed by trackers*** on mobile and desktop environments, and summarized in Table 6 of Appendix C. We found that *window.document.cookie* is the most commonly accessed JavaScript API with 67.4% and 67.5% trackers on mobile and desktop environments using it. Meanwhile, *window.navigator.userAgent* is at second place with 59.8% and 55.4% trackers on mobile and desktop environments using it. The two lists contain the same set of top 10 JavaScript APIs, and their ranks of the top five APIs are even identical. In addition, the percentages of trackers on these two environments that use a corresponding API are close to each other. We did not observe any JavaScript API that is only accessed on the mobile environment. Even for sensor access, the *window.addEventListener* API is used to listen to events such as *devicemotion*, but this API is also widely used to listen to other events on both mobile and desktop environments. However, there are two APIs (i.e., *window.navigator.plugins* and *window.navigator.mimeTypes*) being only accessed on the desktop environment due to the lack of support on the mobile environment.

The results of tracker distribution and tracking JavaScript APIs in this subsection have the following ***major implications***. First, from the perspective of the overall volume of trackers, web tracking on the mobile environment is nearly as pervasive as that on the desktop environment, with 4,052 and 5,073 trackers on mobile and desktop environments, respectively. Second, the top trackers have a great impact on web tracking because they are intensively included by first-party websites on both environments. Third, although the trackers on the long tails access a relatively small number of tracking JavaScript APIs, the overall volume of those trackers is large and the impact of them should never be neglected.

5.1.2 Relationship Between Mobile and Desktop Trackers

Figure 5 is a Venn diagram summarizing the relationship between the trackers identified on mobile websites and desktop websites. We found that in total 4,052 different trackers appeared on the mobile websites and

5,073 different trackers appeared on the desktop websites, indicating the prevalence of including a variety of trackers by first-party websites. These two environments share 3,290 common trackers. We found that 3,107 out of these 3,290 common trackers appeared on matched desktop and mobile first-party websites. That is, each of those trackers appeared on both the mobile and desktop versions of at least one same first-party website. The lists of these trackers can be accessed at a GitHub public repository[7].

Meanwhile, 762 trackers are unique on the mobile environment, indicating that ***a large number of mobile-specific trackers exist on the Internet*** although they only use a subset of JavaScript APIs that appeared on the desktop environment as shown in Table 6. Note that due to the nature of measurement study, these 762 trackers should only be considered as mobile-specific in the results obtained by this study. They may appear on the desktop environment in other studies, and new mobile-specific trackers may be discovered at different times.

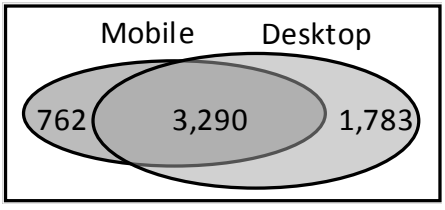


Fig. 5. The number of JS-trackers that appeared on the mobile websites and the desktop websites.

5.1.3 Who and Where Are Those JS-trackers?

To have a better understanding of the 762 mobile-specific trackers in terms of their organization and country information, we leveraged the CrunchBase database [45], Tim Libert's library [57], TLS certificate, and WHOIS record [59] as data sources [8]. In more details, we attempted to identify tracker information from these four data sources in order. For each tracker, we stopped the identification process once the information is obtained from a data source. In total, we successfully retrieved the organization information of 411 trackers from either the CrunchBase database, Tim Libert's library, or TLS certificate, and retrieved the organization information of 251 trackers from the WHOIS record. Figure 13 in Appendix D details the tracker coverage of the four data sources.

Overall, we identified 382 unique organizations that host these trackers. Note that by comparing with the 742 unique organizations hosting the 1,783 desktop-specific trackers, we found that 258 of these 382 organizations only appear on the mobile environment. We summarized the top 10 organizations that host the most number of mobile-specific JS-trackers as shown in Table 2. We identified that five organizations, which are *Adobe*, *Optimzely*, *CloudFlare*, *Amazon*, and *Kameleoon*, provide their content delivery network (CDN) services to various trackers, thus making them ranked in this top 10 list. Interestingly, we found that the rest five are all domain management organizations. They provide domain information protection services, which are privacy measures implemented by a large number of domains. In more details, among those 251 trackers whose organization information was retrived from the WHOIS record, there are 34, 25, 14, 8, and 7 having organization of "Redacted for Privacy", "Domains by Proxy", "Whois Guard", "Global Domain Privacy Services", and "Whois Privacy", respectively. Therefore, these five organizations cannot represent the real organizations of those trackers. These results indicate that ***some trackers tend to use CDN services to host their tracking services and intentionally hide their domain registration information***.

Table 2. Top 10 organizations that host the most number of mobile-specific JS-trackers.

Organization	# of mobile-specific trackers
Adobe	48
Redacted For Privacy	34
Domains By Proxy	25
Optimizely	19
CloudFlare	17
Whois Guard	14
Amazon	9
Kameleoon	9
Global Domain Privacy Services	8
Whois Privacy	7

Table 3 shows the top 10 countries, which are also extracted from the aforementioned data sources, that have the most number of mobile-specific trackers. We found that the *United States* is the top one country that has the most, i.e., 264, mobile-specific trackers.

7 https://github.com/jun521ju/PETS2020_Web_Tracking.

8 D&B Hoovers can also be used for this purpose [26]. We did not leverage it because we could not obtain its database access.

Table 3. Top 10 countries that have the most number of mobile-specific JS-trackers.

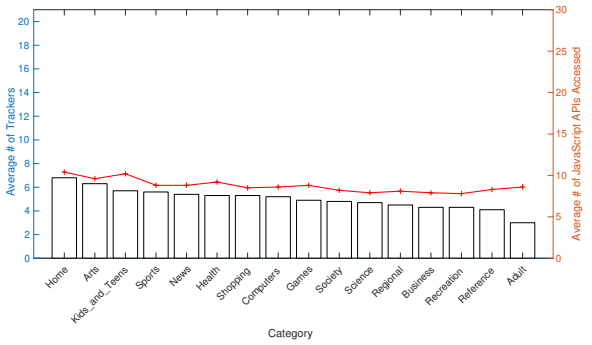
Country	# of trackers	Country	# of trackers
United States	264	Panama	25
China	47	Germany	21
Japan	40	Unite Kingdom	17
France	33	Korea	16
Redacted For Privacy[9]	31	Canada	14

5.1.4 First-party Sites That Include JS-trackers

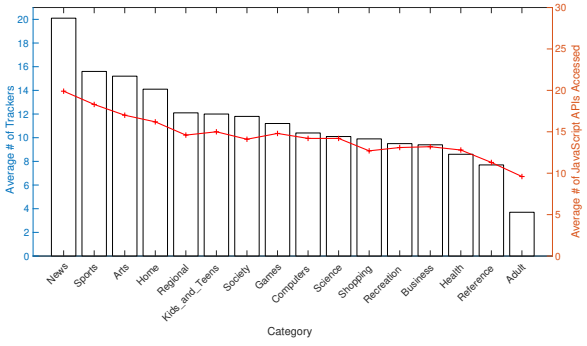
To explore the probability for first-party websites to have more JS-trackers on their desktop versions than on their mobile versions, we performed the *Binomial Proportion statistical test* for the 23,310 first-party websites. Initially, we set our null hypothesis as $H_0 : P = 0.5$, and set our alternative hypothesis as $H_1 : P > 0.5$, where P represents the probability for first-party websites to have more JS-trackers on their desktop versions than on their mobile versions. We increased the value of P with the step of 0.05 to revise our initial hypotheses, and found that we can even reject the null hypothesis with $P = 0.7$ at the 95% confidence interval, which indicates that ***the probability for first-party websites to have more JS-trackers on their desktop versions than on their mobile versions is greater than 0.7***.

To measure the prevalence of trackers included in different categories of the first-party websites, we used Alexa's top 500 sites list in each of the 16 categories to determine the category of each of the first-party websites (note that 1,154 mobile websites and 1,199 desktop websites are successfully categorized because Alexa only provides top 500 sites per category). In Figures 6a and 6b, we then averaged the number of trackers included by the first-party websites in each category represented by the vertical bars with the y-axis on the left side, and averaged the number of JavaScript APIs accessed by the corresponding trackers represented by the curve with the y-axis on the right side.

We found that ***the news category has the most trackers on the desktop environment, while the home category has the most trackers on the mobile environment***. Except for the *adult* category, the rest 15 categories all have nearly double or over double numbers of trackers on desktop websites than on mobile websites; this category-based result is consistent with the aforementioned Binomial Proportion test result. Correspondingly, more tracking JavaScript APIs are accessed on the desktop environment than on the mobile environment.



(a) Mobile websites



(b) Desktop websites

Fig. 6. Average number of trackers appeared on first-party websites of different Alexa categories.

We further explored the prevalence of the trackers on different ranks of the first-party websites as shown in Figure 7. According to Alexa's ranked site list, we first divided the 23,310 websites into six ranking groups. We then averaged in each group the number of trackers (represented by the vertical bars with the y-axis on the left side) included by the first-party websites as well as the number of JavaScript APIs accessed (represented by the curve with the y-axis on the right side) by the corresponding trackers. We found that both the mobile and desktop websites ranked between 1,000 and 5,000 have the most (i.e., 5 and 9, respectively) trackers on average. Generally speaking, the average number of trackers in each group smoothly decreases as the rank decreases except for the ranking group *[1, 1k)*. The *selected* group (i.e., the sites with the rank between 200,000 and 1,000,000 as described in Section 4.1) has the least number of trackers. These results indicate that ***web tracking tends to occur more intensively on the higher ranked websites, no matter for the mobile or desktop environment, than on the lower ranked ones***.

9 The country information is hidden by this domain management organization.

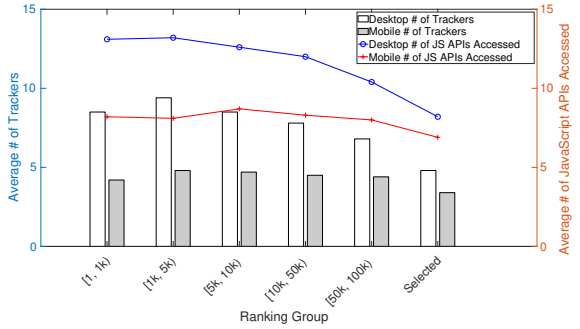


Fig. 7. Average number of trackers included by first-party websites of different Alexa ranking groups on both mobile and desktop environments.

When inspecting the average number of JavaScript APIs accessed in different ranking groups, we found that the number on the desktop environment decreases from 13 in the ranking group *[1, 1k)* to 8 in the *selected* group, while the number on the mobile environment does not significantly change across different ranking groups. On average, the number of JavaScript APIs accessed on mobile and desktop websites is 8 and 12, respectively.

5.1.5 Sensor Access

Motion sensors such as accelerometers and gyroscopes are widely equipped on modern smartphones. Researchers have recently presented motion sensor based device fingerprinting attacks [4, 6, 8, 9] as reviewed in Section 2. For example, the *devicemotion* event in browsers is fired at a regular interval and provides information about the force of acceleration and the rates of rotation of the device. By listening to *devicemotion* events using the window.addEventListener API, attackers can collect motion sensor data from users' devices without their awareness.

As described in Section 3, we instrumented the *window.addEventListener* API to capture all the event handler registration activities. In our measurements, we captured 456 and 302 unique types of events on desktop and mobile environments, respectively, while 212 unique types of events appeared on both environments. Specifically, six types of events appeared on the top 10 list of each environment as shown in Table 4. We observed that events related to sensor access, such as *deviceorientation* and *devicemotion*, only appeared on the mobile environment. In more details, we found that 3.3% (780 out of 23,310) mobile websites accessed the motion sensor data. In addition, we found that 31 mobile websites accessed the ambient light sensor data.

5.2 Tracking Based on HTTP Cookies

While HTTP cookies can be set by JavaScript code, traditionally the majority of cookies are directly set by a browser based on the HTTP responses. What we present in this subsection includes both cases and is a combined result of cookie usage. Since some cookies are placed for functionality purposes instead of tracking, we follow the same method as in Section 4 of [11] to identify ID cookies. ID cookies store unique user identifiers and can be used for tracking. In more details, an ID cookie, which is represented by a tuple of (cookie-name, parameter-name, parameter-value), meets the following four criteria [11]: (1) has a lifetime longer than 90 days, (2) the length of its parameter-value is larger than 7 and less than 101, (3) its parameter-value remains the same in our data collection experiment, and (4) the similarity of its parameter-value between our Experiments A and B is lower than 66% according to the Ratcliff-Obershelp algorithm[55]. We present our analysis based on all the identified ID cookies.

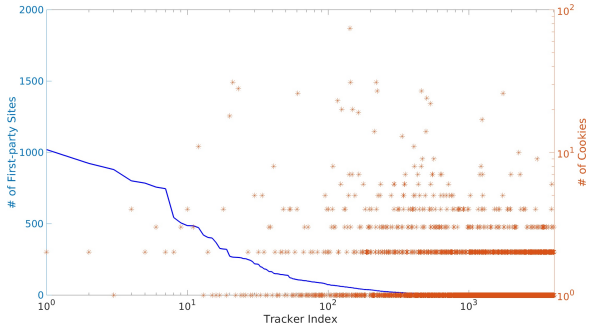
5.2.1 Trackers That Use HTTP Cookies

Figure 8 presents the general view of all the cookie-trackers, i.e., 4,038 on the mobile environment and 4,879 on the desktop environment, in terms of the number of first-party sites they appeared on and the number of the unique cookies they placed. We found the vast majority of these trackers, i.e., 3,540 out of 4,038 trackers on the mobile environment and 4,148 out of 4,879 trackers on the desktop environment, only appear on less than ten first-party websites. The cookies placed by all these trackers vary from 1 to 247. On average, trackers placed cookies on 10 mobile websites and 16 desktop websites.

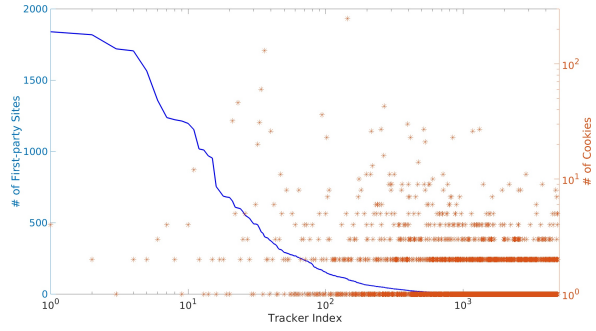
Figures 9a and 9b show the top 20 trackers that placed cookies on the most number of those 23,310 first-party mobile version and desktop version websites, respectively. The y-axis on the left side and the vertical bars represent the number of first-party websites that included the cookie-trackers shown on the x-axis, while the y-axis on the right side and the curve represent the number of different ID cookies created by each of those trackers. Tracker *agkn.com* creates four different cookies on the most number of (i.e., 1,840 or 7.9% of 23,310) first-party desktop websites, while tracker *ad-*

Table 4. Top 10 types of events captured via *window.addEventListener* on 23,310 websites on desktop and mobile environments. The category of *Standard* indicates that an event is defined in some official web specification, and the category of *Framework* indicates that an event is designed in some web development frameworks. Besides, we found that there are roughly 10%, 40%, and 50% of captured events having the event category of *Standard*, *Customized* (i.e., a customized event of a website), and *Framework*, respectively.

Desktop			Mobile		
Event Type	# of Sites	Category	Event Type	# of Sites	Category
scroll	14,168	Standard	scroll	10,662	Standard
message	12,959	Standard	message	9,972	Standard
DOMContentLoaded	5,196	Standard	resize	4,946	Standard
AboutNetErrorOptions	3,277	Framework	DOMContentLoaded	2,333	Standard
pageshow	3,210	Standard	pageshow	2,219	Standard
resize	1,531	Standard	focus	1,010	Standard
startbanneranimations	1,022	Framework	deviceorientation	729	Standard
focus	622	Standard	devicemotion	304	Standard
WebComponentsReady	609	Framework	lazybeforeunveil	164	Framework
adinitialized	594	Framework	error	143	Standard

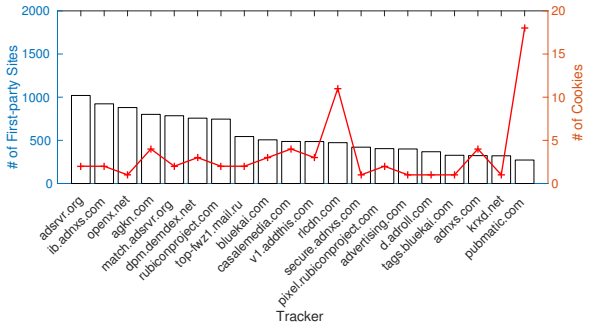


(a) Trackers on the mobile version first-party websites

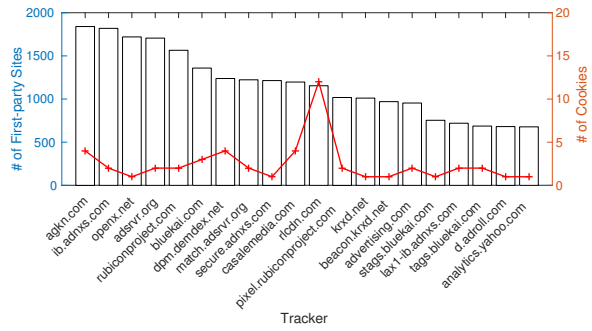


(b) Trackers on the desktop version first-party websites

Fig. 8. All the cookie-trackers that appeared on the most number of those 23,310 first-party websites.



(a) Trackers on the mobile version first-party websites



(b) Trackers on the desktop version first-party websites

Fig. 9. Top 20 trackers that placed cookies on the most number of those 23,310 first-party websites.

srvr.org creates two different cookies on the most number of (i.e., 1,020 or 4.4% of 23,310) first-party mobile websites. With further inspection, we found that both *agkn.com*, owned by the telecommunication company *Neustar Inc.*, and *adsrvr.org*, owned by the online advertising company *The Trade Desk Inc.*, placed cookies for the purpose of targeted advertising according to CookiePedia [44]. Among these top 20 cookie-trackers, the mobile and desktop environments share 16 of them.

As shown in Figure 10, we identified that 4,038 and 4,879 trackers placed cookies on the mobile websites and desktop websites, respectively; meanwhile, we identified that 3,343 trackers placed cookies on both the mobile websites and desktop websites. After comparing these results to the results shown in Figure 5, we found that (1) 672 trackers leverage both JavaScript APIs and HTTP cookies to track users on both the mobile websites and desktop websites; (2) 55 trackers leverage both JavaScript APIs and HTTP cookies to track users specifically on mobile websites; and (3) 176 trackers leverage both JavaScript APIs and HTTP cookies to track users specifically on desktop websites.

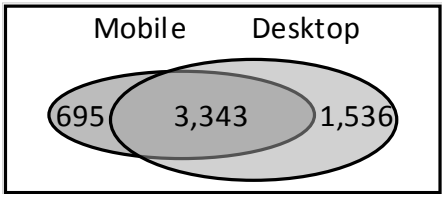


Fig. 10. The number of cookie-trackers that appeared on the mobile websites and the desktop websites.

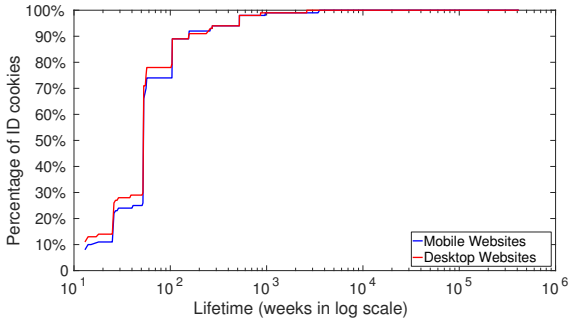


Fig. 11. The lifetime of the cookies placed by trackers on the first-party desktop websites and first-party mobile websites.

Similar to what we did in Section 5.1, we also analyzed the organization and country information of those 695 mobile-specific cookie-trackers leveraging the same four data sources. We found that *Adobe* is the number one organization that hosts 109 mobile-specific cookie-trackers (most of which are using *Adobe's* cloud services, such as *omtrdc.net* and *demdex.net*). Similar to what we observed in Table 2, four of the top ten organizations are domain management service providers. Identical to what we observed in Table 3, *United States* is the top one country that has 315 mobile-specific cookie-trackers.

We further investigated the organizations of all cookie-trackers on both mobile and desktop environments. We identified 278 unique organizations that host 695 mobile-specific cookie-trackers, and 463 unique organizations that host 1,536 desktop-specific cookie-trackers. We found that 211 organizations only appear on the mobile environment, while 396 organizations only appear on the desktop environment. We checked the overlap between the 211 mobile-specific organizations of cookie-trackers and the 258 mobile-specific organizations of JS-trackers (in Section 5.1.3), and found that only 19 organizations are shared between these two sets. For the 239 mobile-specific organizations of JS-trackers that did not place cookies in our measurement experiments, their JavaScript API usage is similar to that of all trackers on the mobile environment as shown in Table 6; note that 150 of these 239 organizations leveraged the *window.document.cookie* API to read cookies.

5.2.2 First-party Sites That Include Cookie-trackers

Similarly, to explore the probability for first-party websites to have more cookie-trackers on their desktop versions than on their mobile versions, we performed the Binomial Proportion statistical test for the 23,310 first-party websites. Using the method as in Section 5.1.4, we can reject the null hypothesis with $P = 0.7$ at the 95% confidence interval. This indicates that ***the probability for first-party websites to have more cookie-trackers on their desktop versions than on their mobile versions is greater than 0.7***.

We further investigated the lifetime of the cookies placed by trackers on the first-party desktop websites and first-party mobile websites. Note that all the ID cookies have a lifetime longer than 90 days as we followed the four criteria in Section 4 of [11] to identify them. Figure 11 illustrates the CDF distribution of the lifetime of all ID cookies. We found that over 70% cookies placed by trackers on both first-party desktop and mobile websites have a lifetime longer than 52 weeks. This means that those cookies can be used by trackers to identify users for a long period of time. Meanwhile, there is no significant difference between the distribution of the lifetime of cookies on mobile websites and that on desktop websites. In addition, we found that over 95% cookies are placed by trackers under the root path (i.e., "/") of their website directories. This implies that those cookies will be carried back in all the requests to the trackers to achieve the maximum tracking effect.

5.3 Summary of Results and Implications

Overall, we found that mobile web tracking has its unique characteristics especially due to mobile-specific trackers. From the perspective of JavaScript APIs, we identified that (1) 762 (13.1%) trackers are mobile-specific, (2) 1,783 (30.6%) trackers are desktop-specific, and (3) 3,290 (56.3%) trackers appear on both mobile and desktop websites. We investigated the organizations of all the mobile-specific trackers, and found that the top ten organizations are either CDN service providers or domain management service providers. We identified 258 organizations that only host JS-trackers for mobile websites. We further analyzed the first-party websites from the perspectives of site category and ranking, and found that (1) the desktop websites of the *news* cate-

gory and the mobile websites of the *home* category include the most trackers, and (2) the average number of trackers in each ranking group smoothly decreases as the rank decreases. From the perspective of HTTP cookies, we identified that (1) 695 (12.5%) trackers are mobile-specific, (2) 1,536 (27.6%) trackers are desktop-specific, and (3) 3,343 (59.9%) trackers place cookies on both mobile and desktop websites. We found that the top ten organizations of the mobile-specific cookie-trackers are either CDN service providers or domain management service providers. We identified 211 organizations that only host cookie-trackers for mobile websites.

The first implication of our comparative measurement study is that web tracking on the mobile environment is increasingly as prevalent as that on the desktop environment in terms of the overall volume of trackers appeared on each environment. However, ***the potential impact of mobile web tracking is more severe*** than that of desktop web tracking because a user may use a mobile device frequently in different places and be continuously tracked. Second, the fact that top trackers appear on a large number of first-party websites implies that those top trackers have a great impact on web tracking. Therefore, web users, developers, and researchers should pay more attention to those top trackers. Third, although the trackers on the long tails access a relatively small number of tracking JavaScript APIs, the overall volume of those trackers is large. Therefore, the impact of those trackers on the long tails should never be neglected either. Last, while trackers can easily collect information of web users, it is very difficult for web users to figure out who are some of those trackers. One reason accounts for this information asymmetry is that some trackers intentionally hide their organization information from the public.

6 Suggestions

Based on our measurement results and analysis, we now tentatively give some suggestions to web users, developers, and researchers to defend against web tracking.

6.1 Suggestions to Web Users

To protect web users from web tracking, our first suggestion is to enhance their awareness of the corresponding privacy concerns. This is essential since web tracking is closely related to each individual web user. Web users should be educated or informed (e.g., via online forums, seminars, social media, etc.) about the privacy concerns lying in web tracking. Note that due to the portability of mobile devices, mobile web users can be tracked all the time when they browse the web. Hence a better awareness of privacy concerns is very important to mobile web users. Our second suggestion is for privacy-conscious web users to adopt adblockers, which can potentially block the data collection from a large volume of trackers that are observed on both mobile and desktop environments as described in Sections 5.1 and 5.2. According to a report from PageFair [36], 11% of the global Internet population was using adblockers on the web as of December 2016. Web users on the desktop environment can either install and configure adblock extensions (e.g., AdBlock [40] and Adblock Plus [41]) on browsers, or use full-fledged browsers such as Cliqz [43] and Brave [42] that block trackers. Many of those adblock choices are also available to mobile web users. For example, mobile web users can also install adblock extensions in their Firefox and Safari browsers, and install the mobile version of Cliqz and Brave browsers. However, mobile Google Chrome does not support extensions even though it shares 55.94% of the mobile browser market by August 2018 according to Statista [37]. To deal with this issue, we suggest mobile web users to install and configure adblock apps (e.g., Disconnect Basic [46], or Adblock Plus for Android that requires a root permission or a proxy to function and needs a user to enable its tracker-blocking option) on their mobile devices, which can help block advertisements on both browsers and apps.

6.2 Suggestions to Developers

For those first-party websites that unintentionally include trackers, we suggest their developers to (1) follow the secure development practices, for example, by complying with the least privilege principle [28] and avoiding the inclusion of extra tracking functions that are beyond the need; (2) localize the inclusion if possible, for example, by copying external JavaScript files and hosting them on the first-party server if allowed (e.g., under the GNU General Public License) to reduce the chance for sending tracking information back to third-party services. For those first-party websites that intentionally include trackers such as advertisement sites, we suggest their developers to (1) figure out what types of information are collected by the trackers before the inclusion, and (2) declare clearly in their privacy policy about the

information of trackers. Such information could better inform web users about third-party web tracking occurred on the first-party websites. A good example is the privacy policy of nytimes.com. It declares the information about trackers by saying: "These companies place or recognize cookies, pixel tags, web beacons or other technology to track certain information about our NYT Services website users" [54]. It also explains that the purpose of tracking could be for serving advertisements by providing some examples.

Besides the above practical suggestions for developers, we believe that an essential and critical suggestion to developers is to act in accordance with the professional ethics. Different codes of conduct have been proposed over the years to inspire and guide developers for ethical decision-making. For example, *ACM Code of Ethics and Professional Conduct* states that "A computing professional should respect privacy" [38], and *International Standard for Professional Software Development and Ethical Responsibility* suggests that "Software engineers shall work to develop software and related documents that respect the privacy of those who will be affected by that software" [52]. In order to defend against web tracking and protect users' privacy, developers should commit themselves to follow these ethics principles. In particular, web developers should identify trackers and strive to minimize the risk of inclusion.

6.3 Suggestions to Researchers

As a popular approach to defending against web tracking, adblockers often filter trackers by using blacklists. All trackers identified in our study are in some blacklists and can thus be blocked. However, maintaining those blacklists, in which most trackers appear on the long tails as we observed in Sections 5.1 and 5.2, requires significant manual effort. Researchers in [33] have shown that the blacklist approach has the problems of insufficient tracker coverage and coarse-grained tracker classification. Therefore, we suggest researchers to further investigate efficient, automated, and fine-grained techniques to identify and update the blacklists of trackers. For example, following the similar direction as [16], researchers could leverage machine learning techniques to develop advanced tracker detection frameworks and improve the performance of adblockers. Our next suggestion to the research community is on developing web tracking visualization tools. Researchers have proposed multiple ways to detect web tracking; however, no work has emphasized web tracking visualization to our best knowledge. A tracking visualization tool that presents intuitive tracking behaviors in real time could help web users become better aware of web tracking and its privacy concerns. Last, researchers have proposed many defense techniques against web tracking such as in [1, 18]. However, those defense techniques are proposed mainly for the desktop environment, so it is unclear whether they will be effective on and compatible with the mobile environment. We therefore suggest researchers to also explore and evaluate defense techniques on the mobile environment (e.g., as in [29]).

7 Conclusion

In this work, we built a framework called WTPatrol that allows us and other researchers to perform web tracking measurement studies on both mobile and desktop environments. Using WTPatrol, we performed the first comparative measurement study of web tracking on 23,310 websites that have both mobile version and desktop version webpages. We conducted an in-depth comparison of web tracking practices on these websites between mobile and desktop environments from two perspectives: web tracking based on JavaScript APIs and web tracking based on HTTP cookies. We identified and analyzed mobile-specific, desktop-specific, and shared JS-trackers as well as cookie-trackers on both environments. Overall, we found that mobile web tracking has its unique characteristics especially due to mobile-specific trackers, and it has become increasingly as prevalent as desktop web tracking in terms of the overall volume of trackers. We tentatively gave some suggestions to web users, developers, and researchers to defend against web tracking. Meanwhile, regulations and laws such as General Data Protection Regulation (GDPR) [51] are with no doubt powerful means to protect users' privacy. We shared the source code of our WTPatrol framework as well as our dataset at a GitHub public repository https://github.com/jun521ju/PETS2020_Web_Tracking.

Acknowledgment

We sincerely appreciate Dr. William Navidi for his valuable suggestions to us on the statistical tests used in this paper, Dr. Nataliia Bielova for her careful shepherding to us, and all anonymous reviewers for their valuable comments and suggestions to us. This research was supported in part by the NSF grant OIA-1936968.

References

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

[2] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. FPDetective: dusting the web for fingerprinters. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2013.

[3] M. Ayenson, D.J. Wambach, A. Soltani, N. Good, and C.J. Hoofnagle. Flash cookies and privacy II: Now with HTML5 and ETag respawning, 2011. http://dx.doi.org/10.2139/ssrn.1898390.

[4] Hristo Bojinov, Yan Michalevsky, Gabi Nakibly, and Dan Boneh. Mobile Device Identification via Sensor Fingerprinting. *CoRR*, abs/1408.1416, 2014.

[5] Qian Cui, Guy-Vincent Jourdan, Gregor V Bochmann, Russell Couturier, and Iosif-Viorel Onut. Tracking phishing attacks over time. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*, 2017.

[6] Anupam Das, Gunes Acar, Nikita Borisov, and Amogh Pradeep. The web's sixth sense: A study of scripts accessing smartphone sensors. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1515–1532. ACM, 2018.

[7] Anupam Das, Nikita Borisov, and Matthew Caesar. Do You Hear What I Hear?: Fingerprinting Smart Devices Through Embedded Acoustic Components. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2014.

[8] Anupam Das, Nikita Borisov, and Matthew Caesar. Tracking Mobile Web Users Through Motion Sensors: Attacks and Defenses. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2016.

[9] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2014.

[10] Peter Eckersley. How Unique is Your Web Browser? In *Proceedings of the International Conference on Privacy Enhancing Technologies (PETS)*, 2010.

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

[12] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W Felten. Cookies that give you away: The surveillance implications of web tracking. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*, 2015.

[13] Christian Eubank, Marcela Melara, Diego Perez-Botero, and Arvind Narayanan. Shining the floodlights on mobile web tracking-a privacy survey. In *Proceedings of the Web 2.0 Security & Privacy (W2SP) Workshop*, 2013.

[14] Seungyeop Han, Jaeyeon Jung, and David Wetherall. A study of third-party tracking by mobile apps in the wild. *Univ. Washington, Tech. Rep. UW-CSE-12-03-01*, 2012.

[15] Thomas Hupperich, Davide Maiorca, Marc Kührer, Thorsten Holz, and Giorgio Giacinto. On the robustness of mobile device fingerprinting: Can mobile users escape modern web-tracking mechanisms? In *Proceedings of the 31st Annual Computer Security Applications Conference*, 2015.

[16] Muhammad Ikram, Hassan Jameel Asghar, Mohamed Ali Kaafar, Anirban Mahanti, and Balachandar Krishnamurthy. Towards seamless tracking-free web: Improved detection of trackers via one-class learning. In *Proceedings on Privacy Enhancing Technologies (PETS)*, 2017.

[17] David M. Kristol. HTTP Cookies: Standards, Privacy, and Politics. *ACM Transactions on Internet Technology*, 2001.

[18] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2016.

[19] Adam Lerner, Anna Kornfeld Simpson, Tadayoshi Kohno, and Franziska Roesner. Internet jones and the raiders of the lost trackers: An archaeological study of web tracking from 1996 to 2016. In *Proceedings of the USENIX Security Symposium*, 2016.

[20] Christophe Leung, Jingjing Ren, David Choffnes, and Christo Wilson. Should you use the app for that?: Comparing the privacy implications of app-and web-based online services. In *Proceedings of the 2016 Internet Measurement Conference*, 2016.

[21] Jonathan R. Mayer and John C. Mitchell. Third-Party Web Tracking: Policy and Technology. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2012.

[22] Keaton Mowery, Dillon Bogenreif, Scott Yilek, and Hovav Shacham. Fingerprinting information in JavaScript implementations. In *Proceedings of the Web 2.0 Security & Privacy (W2SP) workshop*, 2011.

[23] Martin Mulazzani, Philipp Reschl, Markus Huber, Manuel Leithner, Sebastian Schrittwieser, Edgar Weippl, and FC Wien. Fast and reliable browser identification with javascript engine fingerprinting. In *Proceedings of the Web 2.0 Security & Privacy (W2SP) workshop*, 2013.

[24] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2013.

[25] Lukasz Olejnik, Gunes Acar, Claude Castelluccia, and Claudia Diaz. The leaking battery. In *Proceedings of the International Workshop on Data Privacy Management*, 2015.

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

[27] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. Detecting and Defending Against Third-party Tracking on the Web. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2012.

[28] Jerome H Saltzer and Michael D Schroeder. The protection of information in computer systems. In *Proceedings of the IEEE*, 1975.
[29] Anastasia Shuba, Athina Markopoulou, and Zubair Shafiq. Nomoads: Effective and efficient cross-app mobile ad-blocking. *Proceedings on Privacy Enhancing Technologies*, 2018.
[30] Ashkan Soltani, Shannon Canty, Quentin Mayo, Lauren Thomas, and Chris Jay Hoofnagle. Flash Cookies and Privacy. In *Proceedings of the AAAI Spring Symposium: Intelligent Information Privacy Management*, 2010.
[31] Oleksii Starov and Nick Nikiforakis. Xhound: Quantifying the fingerprintability of browser extensions. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2017.
[32] Vincent Toubiana, Arvind Narayanan, Dan Boneh, Helen Nissenbaum, and Solon Barocas. Adnostic: Privacy preserving targeted advertising. 2010.
[33] Zhonghao Yu, Sam Macbeth, Konark Modi, and Josep M Pujol. Tracking the trackers. In *Proceedings of the 25th International Conference on World Wide Web (WWW)*, 2016.
[34] Zhe Zhou, Wenrui Diao, Xiangyu Liu, and Kehuan Zhang. Acoustic fingerprinting revisited: Generate stable device id stealthily with inaudible sound. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2014.
[35] Mobile web browsing overtakes desktop for the first time, 2016. https://www.theguardian.com/technology/2016/nov/02/mobile-web-browsing-desktop-smartphones-tablets.
[36] PageFair, 2017. https://pagefair.com/downloads/2017/01/PageFair-2017-Adblock-Report.pdf.
[37] Market share held by leading mobile internet browsers, 2018. https://www.statista.com/statistics/263517/market-share-held-by-mobile-internet-browsers-worldwide/.
[38] ACM Code of Ethics and Professional Conduct, 2019. https://www.acm.org/code-of-ethics/.
[39] Ad block engine of Brave Browser, 2019. https://github.com/brave/ad-block.
[40] AdBlock, 2019. https://getadblock.com/.
[41] Adblock Plus, 2019. https://adblockplus.org/.
[42] Brave Browser, 2019. https://brave.com/.
[43] Cliqz Browser, 2019. https://cliqz.com/.
[44] CookiePedia, 2019. https://cookiepedia.co.uk/.
[45] CrunchBase, 2019. https://www.crunchbase.com/.
[46] Disconnect Basic, 2019. https://disconnect.me/disconnect.
[47] EasyList, 2019. https://easylist.to/easylist/easylist.txt.
[48] EasyListVari, 2019. https://easylist.to/pages/other-supplementary-filter-lists-and-easylist-variants.html.
[49] EasyPrivacy, 2019. https://easylist.to/easylist/easyprivacy.txt.
[50] Fully Qualified Domain Name, 2019. https://en.wikipedia.org/wiki/Fully_qualified_domain_name.
[51] General Data Protection Regulation, 2019. https://eugdpr.org/the-process/timeline-of-events/.
[52] International Standard for Professional Software Development and Ethical Responsibility, 2019. https://www.etsu.edu/cbat/computing/seeri/ethics-code.php.
[53] Mozilla Public Suffix List, 2019. https://publicsuffix.org/list/public_suffix_list.dat.
[54] Privacy Policy of nytimes.com, 2019. https://help.nytimes.com/hc/en-us/articles/115014892108-Privacy-policy.
[55] Ratcliff/Obershelp pattern recognition, 2019. https://xlinux.nist.gov/dads/HTML/ratcliffObershelp.html.
[56] Selenium Web Driver, 2019. http://www.seleniumhq.org/.
[57] Timothy Libert's Library, 2019. https://github.com/timlib/webXray_Domain_Owner_List.
[58] Web of Trust, 2019. https://www.mywot.com/.
[59] WHOIS, 2019. https://www.whois.com/whois/.
[60] World Wide Web Consortium, 2019. https://html.spec.whatwg.org/multipage/dom.html.

A Proportional Distance Example

Figure 12 shows the DOM trees of two hypothetical example webpages, p_1 and p_2. To calculate the proportional distance between p_1 and p_2, we first define an arbitrary but fixed ordering of a corpus of n HTML tags. For the sake of simplicity, we define a corpus of nine HTML tags with the following fixed order in this example: " , <h1>, , <p>, <form>, <a>, <button>, <script>, <style>". As detailed in Section 4.2, we then construct the vector t_1, i.e., (2, 2, 1, 1, 1, 0, 0, 0, 0), for the webpage p_1, and the vector t_2, i.e., (3, 0, 1, 0, 0, 2, 1, 0, 0), for the webpage p_2. Because there are seven out of nine tags appeared on webpages p_1 and p_2, and six of them appeared the different number of times on both vectors, the accumulated Hamming distance between vectors t_1 and t_2 is six, and the proportional distance between p_1 and p_2 is thus $6/7$, i.e., 0.857, based on Formula 1.

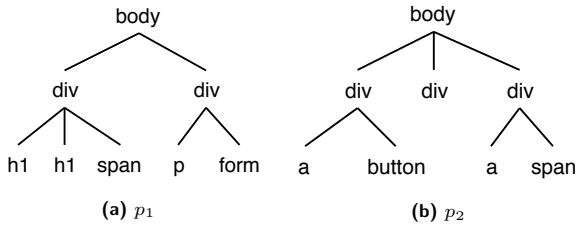


Fig. 12. The DOM trees of webpages p_1 and p_2.

B Manual Inspection Result Between Real and Emulated Mobile Browsers

Table 5. We visited the homepages of the top 100 Alexa websites and compared the corresponding webpage sources between a real mobile Firefox browser and an emulated mobile Firefox browser. Eight websites returned webpages differing in at least one script element as exemplified in this table. The last column presents the potential usage of JavaScript code based on our manual inspection.

Site	Browser	Example Code Difference	Usage of JavaScript Code
so.com	Mobile	<script id="stc_home_next_base"> (function(){"use strict"; function e(e) {return typeof e=="function" || typeof e=="object" && e!==null} ... </script>	Customization of the search input field and function.
	Emulated	No such code.	-
stackoverflow.com	Mobile	<script>StackExchange.using ("gps", function () { StackExchange.gps.track ("homepage.visit", {}, true); });</script>	GPS-based tracking.
	Emulated	No such code.	-
amazon.co.jp	Mobile	<script type="text/javascript"> ue_csm.ue.exec(function(b,e) {function q() ... </script>	Utility function related to XMLHttpRequest.
	Emulated	No such code.	-
booking.com	Mobile	<script> ... b_browser_family: 'Firefox', b_is_android: '1', ... </script>	Browser detection.
	Emulated	<script> ... b_browser_family: 'Mobile Other', b_is_android: '', ...</script>	Browser detection.
amazon.de	Mobile	<script type="text/javascript"> ue_csm.ue.exec(function(b,e) {function q() ... </script>	Utility function related to XMLHttpRequest.
	Emulated	No such code.	-
wordpress.com	Mobile	<script>(function() { var i = new Image()...</script>	Tracking based on a single pixel image.
	Emulated	No such code.	-
xhamster.com	Mobile	<script type="application/javascript" src="https://cdn.tsyndicate.com /sdk/v1/n.js"> </script>	Loading an advertisement realted SDK.
	Emulated	No such code.	-
amazon.in	Mobile	<script> P.declare ('devmea:deviceLatencyData'... </script>	Utility function related to device latency measurement.
	Emulated	No such code.	-

C All the JavaScript APIs Accessed By Trackers

Table 6. All the JavaScript APIs accessed by 5,073 trackers on the desktop environment and by 4,052 trackers on the mobile environment, respectively. There are two APIs (i.e., *window.navigator.plugins* and *window.navigator.mimeTypes*) being only accessed on the desktop environment due to the lack of support on the mobile environment.

Desktop		Mobile	
JavaScript API	# (percent) of trackers	JavaScript API	# (percent) of tracker
window.document.cookie	3,422 (67.5%)	window.document.cookie	2,732 (67.4%)
window.navigator.userAgent	3,049 (60.1%)	window.navigator.userAgent	2,422 (59.8%)
window.localStorage	1,885 (37.2%)	window.localStorage	1,498 (37.0%)
window.Storage	1,844 (36.3%)	window.Storage	1,487 (36.7%)
window.navigator.cookieEnabled	1,347 (26.6%)	window.navigator.cookieEnabled	1,131 (27.9%)
window.performance	1,119 (22.1%)	window.sessionStorage	844 (20.8%)
window.sessionStorage	1,049 (20.7%)	window.performance	818 (20.2%)
window.navigator.language	1,030 (20.3%)	window.navigator.language	803 (19.8%)
window.navigator.platform	816 (16.1%)	window.screen.colorDepth	645 (15.9%)
screen.colorDepth	806 (15.9%)	window.navigator.platform	604 (14.9%)
window.navigator.appName	665 (13.1%)	window.navigator.appName	439 (10.8%)
window.navigator.plugins	538 (10.6%)	window.navigator.doNotTrack	340 (8.4%)
window.navigator.doNotTrack	454 (8.9%)	window.name	293 (7.2%)
window.name	392 (7.7%)	window.navigator.appVersion	290 (7.2%)
window.navigator.appVersion	381 (7.5%)	window.navigator.vendor	226 (5.6%)
window.navigator.vendor	332 (6.5%)	HTMLCanvasElement	196 (4.8%)
HTMLCanvasElement	316 (6.2%)	window.navigator.product	130 (3.2%)
window.navigator.product	182 (3.6%)	window.screen.pixelDepth	117 (2.9%)
window.addEventListener	181 (3.6%)	window.addEventListener	111 (2.7%)
CanvasRenderingContext2D	163 (3.2%)	CanvasRenderingContext2D	109 (2.7%)
window.screen.pixelDepth	153 (3.0%)	window.navigator.oscpu	108 (2.7%)
window.navigator.oscpu	153 (3.0%)	window.navigator.onLine	40 (1.0%)
window.navigator.mimeTypes	138 (2.7%)	window.navigator.appCodeName	35 (0.9%)
window.navigator.onLine	72 (1.4%)	window.screen.orientation	34 (0.8%)
window.navigator.appCodeName	55 (1.1%)	window.navigator.geolocation	34 (0.8%)
window.screen.orientation	51 (1.0%)	AudioContext	20 (0.5%)
window.navigator.geolocation	47 (0.9%)	RTCPeerConnection	15 (0.4%)
AudioContext	28 (0.6%)	OscillatorNode	15 (0.4%)
RTCPeerConnection	22 (0.4%)	window.navigator.buildID	15 (0.4%)
OscillatorNode	22 (0.4%)	GainNode	2 (0.0%)
window.navigator.buildID	20 (0.4%)	ScriptProcessorNode	1 (0.0%)
GainNode	3 (0.1%)	AnalyserNode	1 (0.0%)
AnalyserNode	2 (0.0%)		
ScriptProcessorNode	2 (0.0%)		

D Trackers Coverage by Four Data Sources

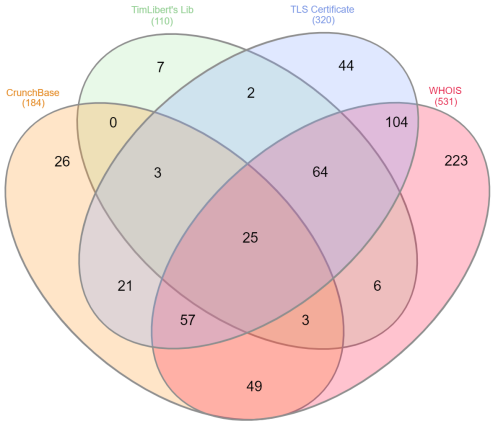


Fig. 13. Organization information of the 762 mobile-specific JS-trackers covered by the CrunchBase database, Tim Libert's library, TLS certificate, and WHOIS record. The four data sources complement each other, but the WHOIS record provides the best coverage.