The Seven Deadly Sins of the HTML5 WebAPI: A Large-scale Study on the Risks of Mobile Sensor-based Attacks

MICHALIS DIAMANTARIS, FORTH, Greece
FRANCESCO MARCANTONI, University of Illinois at Chicago, USA
SOTIRIS IOANNIDIS, FORTH, Greece
JASON POLAKIS, University of Illinois at Chicago, USA

Modern smartphone sensors can be leveraged for providing novel functionality and greatly improving the user experience. However, sensor data can be misused by privacy-invasive or malicious entities. Additionally, a wide range of other attacks that use mobile sensor data have been demonstrated; while those attacks have typically relied on users installing malicious apps, browsers have eliminated that constraint with the deployment of HTML5 WebAPI.

In this paper we conduct a comprehensive evaluation of the multifaceted threat that mobile web browsing poses to users, by conducting a large-scale study of mobile-specific HTML5 WebAPI calls across more than 183K of the most popular websites. We build a novel testing infrastructure consisting of actual smartphones on top of a dynamic Android app analysis framework, allowing us to conduct an end-to-end exploration. In detail, our system intercepts and tracks data access in real time, from the WebAPI JavaScript calls down to the Android system calls. Our study reveals the extent to which websites are actively leveraging the WebAPI for collecting sensor data, with 2.89% websites accessing at least one sensor. To provide a comprehensive assessment of the risks of this emerging practice, we create a taxonomy of sensor-based attacks from prior studies, and present an in-depth analysis by framing our collected data within that taxonomy. We find that 1.63% of websites can carry out at least one attack and emphasize the need for a standardized policy across all browsers and the ability for users to control what sensor data each website can access.

CCS Concepts: • **Security and privacy → Mobile platform security**; Browser security.

Additional Key Words and Phrases: Android; mobile HTML5; WebAPI; mobile sensors; sensor attack taxonomy; browser guidelines

ACM Reference Format:
Michalis Diamantaris, Francesco Marcantoni, Sotiris Ioannidis, and Jason Polakis. 2020. The Seven Deadly Sins of the HTML5 WebAPI: A Large-scale Study on the Risks of Mobile Sensor-based Attacks. *ACM Trans. Priv. Sec.* 1, 1, Article 1 (January 2020), 31 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Mobile phone usage has been on a constant rise, and smartphone devices have reached a near-ubiquitous presence in many countries. According to reports, almost all mobile phone owners in the United States own a smartphone [73]. Amongst the rich set of functionality offered by such devices web browsing remains popular. Smartphone usage has gained such traction, that in 2016 more

Authors' addresses: Michalis Diamantaris, FORTH, Greece, diamant@ics.forth.gr; Francesco Marcantoni, University of Illinois at Chicago, USA, f.marcantoni94@gmail.com; Sotiris Ioannidis, FORTH, Greece, sotiris@ics.forth.gr; Jason Polakis, University of Illinois at Chicago, USA, polakis@uic.edu.

2471-2566/2020/1-ART1 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

internet traffic originated from mobile devices than desktop computers worldwide [57] and 56% of the traffic to top-sites in the United States was from mobile devices [115]. This upward trend has been influenced by many factors, including improvements in the speed of mobile internet connections, mobile browsers offering more features, and improvements in smartphone hardware [50].

On the other hand, apart from the obvious usability benefits, smartphone devices have also introduced a plethora of privacy risks. In this post-Snowden era [58] users are becoming increasingly aware of privacy issues including online tracking and internet surveillance, and employ private browsing among other techniques to remain anonymous online (despite overestimating the protection it actually offers [125]). Nonetheless, private browsing offers some protection against users being tracked across different sessions [126]. However, adversaries can still track users through browser or device fingerprints [93]. This consists of collecting characteristics of the device environment and the browser itself, making it possible to identify which device is navigating a given webpage [71]. Prior work has also shown that manufacturing imperfections in sensors' hardware render them fingerprintable [45].

Websites can access mobile sensor data through the HTML5 WebAPI, which is supported by major browsers (with certain variations). The WebAPI is not limited to mobile devices, and offers a rich set of capabilities to modern websites. Accordingly, Snyder et al. [111] presented a cost-benefit analysis of the functionality of the WebAPI using a small set of websites, by correlating WebAPI calls to vulnerabilities reported in CVE reports and relevant academic attacks (including tracking, cross-origin information stealing [117], and timing attacks [59]). As that study focused on desktop browsing and was limited in scale, Das et al. recently explored the pervasiveness of mobile device fingerprinting [42]. However, this is only one of the threats that the WebAPI poses to mobile users and no comprehensive large-scale exploration currently exists. In practice a plethora of attacks that were previously limited to mobile apps can "migrate" to the mobile web, as modern browsers provide access to a device's underlying mobile sensors.

In this paper we present a *quantitative* and *qualitative* large-scale study of mobile-specific WebAPI calls made by websites in the wild. We build a unique crawling infrastructure that uses real Android devices and perform an end-to-end analysis of WebAPI requests. In more detail, apart from injecting a script in websites that allows us to hook all WebAPI calls, we leverage a dynamic real-time app-analysis system that allows us to trace the internal behavior of the Android OS when calls occur, ensuring the fidelity of our measurements. Using our crawling infrastructure, we measure the prevalence of mobile-specific WebAPI calls across 183,571 of the most popular websites during March-September 2018. Our experiments capture the true scale of this phenomenon, as we detect 5,313 unique domains accessing at least one mobile WebAPI call; 35.89% of those also result in sensors being accessed by third-party scripts that originate from 11 second-level domains. To better understand the implications and potential threat that users face, we survey prior literature on attacks from malicious apps that leverage data from mobile sensors. Based on this diverse yet representative selection of papers we create a taxonomy of attacks that can be potentially carried out by modern websites by obtaining seven categories of sensor data. We then break down the different attacks based on their sensor requirements and conduct an in-depth analysis of our dataset, and find that 3,008 websites request access to sensor data needed for at least one attack. While we refer to attacks, these capabilities include privacy-invasive behavior (e.g., inferring a user's sensitive demographic information for personalizing an ad) that can be carried out by, otherwise, legitimate websites.

Our extensive analysis reveals that popular websites, as well as websites from popular categories (e.g., e-banking), tend to access more mobile sensors which consequently leads to the feasibility of conducting more sensor-based attacks. Interestingly at least one domain from *every* category in our dataset could potentially infer the user's input, which is also the most frequently feasible attack

across all categories and can be used to steal sensitive information (e.g., credit card information and pin number). Moreover websites do not need to access a lot of different sensors to perform these attacks; readings from the motion and orientation sensors, which do *not* require any permission at the operating system level, can lead to 9 and 8 different attacks respectively. We argue that with different browsers enforcing different access policies, as do the plethora of apps that support WebView, there is dire need for a standardized, fine-grained universal mechanism that allows users to control access to *all* types of mobile sensor data.

Overall, this paper makes the following contributions:

- We conduct a large-scale end-to-end study of websites targeting mobile-specific sensors across 183K of the most popular websites according to Alexa, and our findings reveal the extent to which websites target smartphone users. We use a novel crawling infrastructure consisting of smartphone devices, which prevents potential evasive behavior from domains that can infer the presence of a virtualized execution environment (e.g., through canvas fingerprinting).
- We provide a taxonomy of previously reported sensor-based attacks and reframe them within the modern mobile ecosystem where WebAPI and WebView are widely supported and attacks are not constrained to users that install a malicious app. Guided by our taxonomy we conduct a qualitative and quantitative analysis of our collected data and provide a comprehensive assessment of the risks presented by the mobile WebAPI.
- Due to the severity of the attacks enabled by mobile sensors, we provide a set of guidelines for access control policies that should be adopted and standardized across browsers to better protect users and also allow them more control over their data.
- Our dataset is publicly available at https://www.cs.uic.edu/~webapi to further facilitate research on the security and privacy risks that users face due to mobile sensor data being accessible to websites.

2 BACKGROUND

In this section we provide an overview of mobile-specific calls supported by the HTML5 WebAPI and then present a taxonomy of attacks presented in prior work that rely on data provided by mobile sensors.

2.1 HTML5 WebAPI

Browsers have evolved significantly from their original design, both in terms of the functionality they provide as well as their underlying complexity. At the same time, the advent of smartphones and their almost ubiquitous presence has enabled a wide range of previously-infeasible functionality due to connectivity on-the-go and the information returned by embedded mobile sensors. As such, many capabilities previously restricted to native apps are now available to websites. While the WebAPI introduces obvious usability benefits to end users as it can improve the overall experience, it also poses a significant privacy and security risk. Apart from enabling certain forms of user tracking (e.g., through the discontinued Battery API [96]) other sensor-based attacks that were previously restricted to mobile apps can now be deployed over the web. To better explore this threat, we focus on all mobile-specific HTML5 WebAPI calls, and subsequently explore the attacks that they enable. In detail, our study focuses on the following WebAPI calls (which we refer to as mobile-specific for the remainder of the paper).

DeviceMotionEvent.acceleration [22]: This call provides web developers with information from the accelerometer sensor about the speed of changes in the device's position, returning values expressed in m/s^2 for all three X, Y, Z axes.

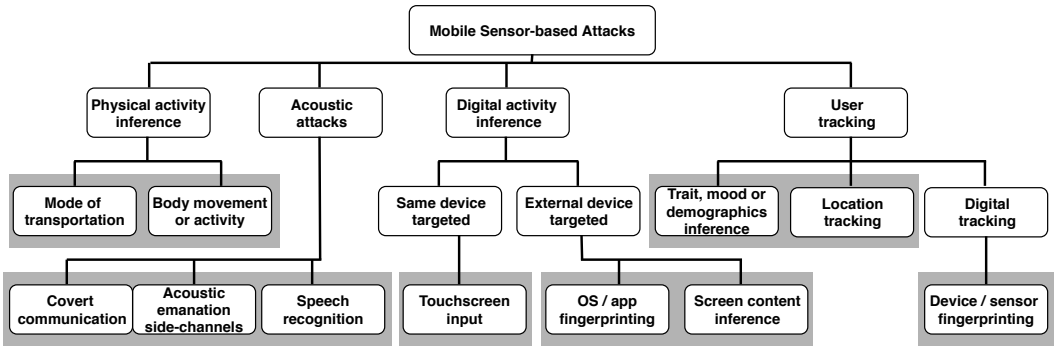

Fig. 1. Taxonomy of attacks demonstrated in prior studies that leverage data from mobile sensors.

DeviceMotionEvent.rotationRate [86]: This call returns information from the gyroscope sensor about the rate at which the device's orientation changes along the three orientation axis (alpha, beta, gamma). This value is expressed in degrees per second.

DeviceOrientationEvent [104]: This event is fired when the accelerometer detects a change to the device's orientation (i.e., from landscape to portrait and vice versa).

DeviceProximityEvent [23]: This allows websites to obtain information about the distance of a nearby physical object using the device's proximity sensor. The value is returned in centimeters. For instance, this information can be used for energy conservation by turning a device's screen off when the user is talking on the phone.

DeviceLightEvent [20]: Websites can obtain information about changes in the device's environment by indicating changes in the intensity of the light as measured by the ambient light sensor, which is expressed in lux units.

Geolocation [19]: This set of API calls allows web developers to retrieve the geographical position of a smartphone device in real time. This is done at two levels of granularity: `fine` which relies on readings for the device's GPS, or `coarse` which relies on information of the WiFi network the device is connected to. Specifically the `getCurrentPosition()` method instantly retrieves the position of the device, while the `watchPosition()` method returns the position of the device using a new a process that continuously polls for the current location.

getUserMedia [38]: This set of calls provides access to the device's camera and microphone sensors.

vibrate() [21]: This allows websites to control the smartphone's vibration engine, and supports different vibration patterns of different durations to be sent to the device.

2.2 Attack Taxonomy

A plethora of research papers have demonstrated mobile-based attacks that employ sensor data. While a considerable number of attacks present similar characteristics, e.g., demonstrating different techniques for inferring a user's touchscreen input or fingerprinting the user's device, a wide range of different attacks have been proposed. Here we introduce a taxonomy of attacks compiled from the literature that captures the vast potential of how the seven different categories of mobile sensor data can be misused by adversaries. Typically these attacks assume that attackers are able to obtain sensor data through a malicious app installed on the device. However, in practice, modern browsers can mediate data exchange between websites and sensor data through the HTML5 WebAPI. This leads to a different threat model and an increased attack surface, as it removes the constraint of users having to install a malicious app; simply visiting a website can expose users to these attacks.

While this might affect the accuracy of certain attacks (e.g., due to a website being able to obtain sensor data for a shorter amount of time compared to an app) it remains an important and largely unexplored risk for mobile users.

In Figure 1 we present our taxonomy which aims to highlight the variety of attacks enabled by sensor-data, while simultaneously obscuring the type of sensor used for each attack. We do not include explicit sensor information in our taxonomy, as prior attacks often obtain the same objective while using different combinations of sensors (as can be seen in Table 1). At the same time we opt for a relatively fine-grained first level, and specifically consider acoustic attacks as a separate class due to their unique and diverse nature, instead of including them as sub-classes of physical and digital activity inference attacks. Next we briefly describe the four main classes from our taxonomy's first level and refer to some of the presented attacks.

Physical activity inference. Numerous studies [60, 64, 80, 90, 102] have demonstrated that mobile sensors can be used to infer information about personal everyday activities. For example it is possible to infer whether the user is walking, running or their mode of transportation, by leveraging the Motion and GPS sensors [102].

Acoustic attacks. [18, 44, 54, 55, 77, 78, 82, 107] showed that access to Accelerometer, Gyroscope or the Vibration API can be used to infer users' credit card numbers by listening for specific frequencies [107] or what a user is typing on a physical keyboard [78], and bypassing dynamic analysis systems and antivirus products through covert channel attacks [77].

Digital activity inference. This class includes a wide range of attacks, with prior work [31, 34, 36, 51, 62, 80, 97, 113, 128] showing that sensor information (including the Accelerometer and Gyroscope) can be used to predict what the user is typing on the smartphone's touchscreen(e.g., [80, 97]). This is possible because typing leads to changes in the position of the screen, its orientation and the device's motion. In a different study, the Light sensor was used to identify the content of an external display and even classify users' digital activities into different categories with an 85% accuracy [36].

User tracking. Identifying and tracking users across the web has garnered much attention [16, 17, 32, 40–43, 45, 51, 60, 65, 66, 82, 90, 102, 103, 131, 133]. This can be conducted in different ways, from coarse-grained location tracking that does not require any user-permission (using just the Accelerometer or Gyroscope) [60, 90], to fine-grained device fingerprinting using rich and high-resolution data from smartphone sensors(e.g., [17, 45]). In this category we also include alternative attacks that could track users by inferring demographic information (e.g., age [43], gender [82] and fingerprints [51]), physical traits such as their gait [66, 103], or information about their mental state or mood [133].

Deconstructing sensor attacks. Table 1 lists the different sensor-based attacks previously described in the studies that guided our taxonomy. We classify previous attack papers based on the taxonomy introduced in Figure 1. Subsequently, we break down all the attacks presented in those papers based on the type of sensor data needed to carry out the attacks. If an attack can be carried out using a single sensor, that sensor is denoted with ●. For attacks that require multiple sensors to succeed we mark the sensors with ◖ . For sensors that are not required, but can be used to improve accuracy we use ○. For example the technique in [102] that infers the body movement or activity of a user requires access to the motion and GPS sensors. On the other hand, [90] only requires the orientation sensor, but using the Motion or Magnetometer sensor can further improve the attack. Even though access to the magnetometer is not currently supported by Firefox [4], which we used, we include it for completeness.

3 METHODOLOGY AND SYSTEM DESIGN

In this section we present our system design and experimental methodology. We give an overview of our system's architecture, and provide implementation details about the in-line hooking methods

Table 1. Sensor based side-channel attacks.

Ref.#	Attack	Motion	Orientation	Light	Vibration	Magnetometer	Media (Camera,Mic)		GPS
[102]	Mode of transportation	◖	-	-	-	-	-	-	◖
[60, 61, 64]		●	-	-	-	-	-	-	-
[92]		●	-	-	-	●	-	-	-
[90]		○	●	-	-	○	-	-	-
[124, 129]		◖	◖	-	-	◖	-	-	-
[116]		●	●	-	-	-	-	-	-
[102]	Body movement or activity	◖	-	-	-	-	-	-	◖
[80, 129]		◖	◖	-	-	-	-	-	-
[69]		●	-	-	-	-	-	-	-
[116]		●	●	-	-	-	-	-	-
[63, 102]	Location Tracking	◖	-	-	-	-	-	-	◖
[60]		●	-	-	-	-	-	-	-
[90]		○	●	-	-	○	-	-	-
[91]		◖	◖	-	-	-	-	-	-
[124]		◖	◖	-	-	◖	-	-	-
[92]		●	-	-	-	●	-	-	-
[78]	Acoustic emanation side-channels	●	-	-	-	-	-	-	-
[54, 55, 127, 132]		-	-	-	-	-	-	●	-
[82]	Speech recognition	-	●	-	-	-	-	-	-
[18]		◖	◖	-	-	-	-	-	-
[34, 62, 80, 84, 99, 128]	Touchscreen input	◖	◖	-	-	-	-	-	-
[97],[24],[35]		●	-	-	-	-	-	-	-
[35]		-	●	-	-	-	-	-	-
[113]		-	-	●	-	-	-	-	-
[51, 100]		-	-	-	-	-	●	-	-
[108]		◖	-	-	-	◖	-	-	-
[89]		-	●,○	-	-	-	-	●,○	-
[109]		-	-	-	-	-	◖	◖	-
[31]	OS/app fingerprinting	-	-	-	-	●	-	-	-
[36]	Screen content inference	-	-	●	-	-	-	-	-
[43, 103, 133]	Physical trait or demographics inference (e.g., age, sex, mood, fingerprints, gait)	●	-	-	-	-	-	-	-
[33]		●	-	-	-	-	-	-	-
[82]		-	●	-	-	-	-	-	-
[51]		-	-	-	-	-	●	-	-
[53, 66]		◖	◖	-	-	-	-	-	-
[70]		-	-	-	-	-	-	●	-
[41, 42, 65, 129]	Device/sensor fingerprinting	◖	◖	-	-	-	-	-	-
[32]		●	-	-	-	-	-	●	-
[17]		●,◖	●	-	-	-	◖	-	-
[45]		●	-	-	-	-	-	-	-
[40, 131]		-	-	-	-	-	-	●	-
[16]		◖	◖	-	-	◖	-	◖	-
[77]	Covert communication side-channels	-	-	-	●	-	-	-	-
[107]		-	-	-	-	-	-	●	-
[44]		◖	-	-	◖	-	-	-	-
[130]		●	●	-	-	-	-	-	-

Sensors marked with (●) are sufficient for performing the specific attack. When a combination of multiple sensors is required to perform the attack, they are marked with (◖). We denote optional sensors with (○) (e.g., that data is optional and enhances the accuracy of the attack). When papers present multiple attacks, combinations of all of the above may be present in the table. Grey columns denote sensor data that should require explicit user permission according to the W3C.

for intercepting both JavaScript and Android system call functions. Due to space constraints we omit certain details about our system's functionality regarding the Android internals.

System architecture. Our system employs a transparent proxy server that intercepts network traffic by using `mitmproxy` [37]. We configured all the Android devices used in our experiments with `mitm`'s certificate in order to intercept both HTTP and HTTPS traffic. As can be seen in

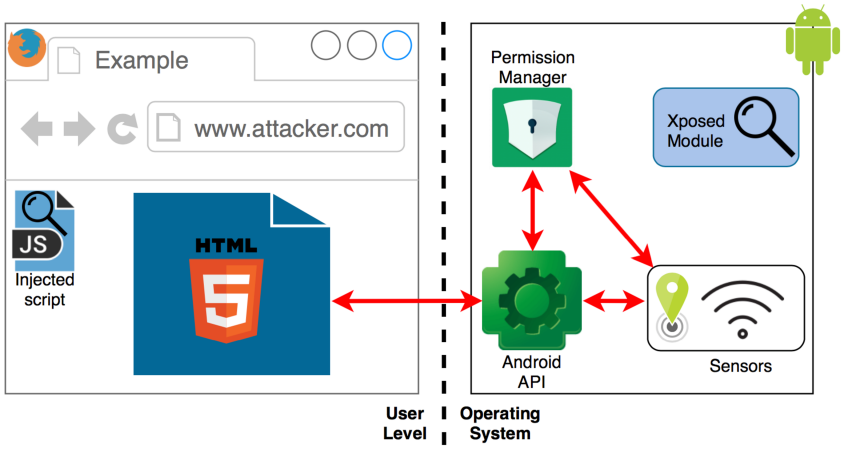

Fig. 2. Overview of our crawling system's architecture. Our system components provide an end-to-end view of requests to access the mobile sensors. The red arrows denote communication "pathways" observed by our system.

Figure 2, the proxy server injects a JavaScript component that hooks and monitors JavaScript calls to mobile-specific WebAPIs. However, our aim is to obtain an in-depth view of sensor data access. In general, browsers are responsible for mediating access between high-level JavaScript function calls and low-level Android API calls. Understanding how this mechanism works for every browser would be time consuming and in many cases infeasible due to proprietary code. As such, we have opted for a generic and browser-agnostic approach, where we intercept Android system calls using a custom module for the Xposed framework [105] that (i) detects and hooks requests to sensor-specific Android API calls and (ii) identifies which of these API calls are permission-protected through the list of permission protected calls found in [2] (such requests are handled by the Permission Manager). By intercepting these low-level function calls we are able to validate that the JavaScript interception was successful and calls requesting sensor data were correctly logged.

Mobile HTML5 Functions. We identified the functions that retrieve mobile-specific data through the official mobile HTML5 WebAPI [52], which lists all the available features. We consider as *mobile-specific* any calls that obtain information originating from an integrated sensor of a mobile device. The HTML5 WebAPI calls interact with the webpage using either a direct one-time communication (i.e., Vibration and Media capture) or through an event listener, since some sensors (i.e., Motion, Orientation, Proximity, and Ambient Light) continuously fire events in order to provide up-to-date readings in real time. For Geolocation one call exists for each category. The full list of identified mobile HTML5 functions maps to seven mobile sensors, as shown in Table 1.

JavaScript Calls Interception. We build our component for hooking JavaScript methods upon the `javascript-hooker` Node.js module [30]. This script allows us to hook any JavaScript function called in a webpage, and passes as arguments the actual `object`, the `method` to hook and the `function` that will be called before the execution of the hooked method. The script creates a wrapper around the functions and substitutes the original method. In our experiments we do not overwrite the original function but only need to identify whether a function is called. Thus, whenever a mobile HTML5 WebAPI is called the JavaScript modules creates a log entry for further analysis and executes the original function. Since `javascript-hooker` also takes the arguments of the original function, we can also intercept the arguments of the `addEventListener` and check for

events of interest. Our code is directly injected in the `head` of the document (if there is one) or the page `body` otherwise.

In order to listen to events and associate a function to a specific target we need to intercept the `setter` property. Even though this is possible using `Object.defineProperty()` the original value will be lost and the webpage may not function as expected. Therefore, we follow the approach employed by Chameleon [56] and overwrite the `getter` property of each event prototype instead of substituting the setter relative to the target property. As such, every time the property is read, our custom function is called. While certain sensor data may normally remain the same during navigation (e.g., properties related to display characteristics), remaining constant might be considered "suspicious" for other sensors. For instance, when an actual human uses the device, small changes in the gyroscope readings would be expected. As such, to make our crawling more realistic, our system intercepts the values returned by certain sensors and slightly modifies their value while ensuring that the result remains "valid"; For instance, a `gyroscope` orientation reading is a decimal number between −365 and +365. In general, data retrieved through events, is handled in two different ways: listening to `addEventListener` on the target object while checking if the argument matches the desired event and defining new `getters` for the properties of the event's prototype.

Identifying the JavaScript source. Apart from logging WebAPI calls we also want to identify the origin of the JavaScript files being executed. This information is important in order to identify if the script belongs to a first-party domain or a third-party domain. We register the source of the URL by utilizing the `stack` property of the `Error` object. Our hooking script implements a mechanism that creates an `Error` object and reads its `stack` property.

Android API call interception. Each mobile HTML5 WebAPI is associated with a low-level Android API call. In order to validate the results of the JavaScript interception and to identify which ones require a permission, we use the PermissionHarvester [46] module that hooks every Android permission protected API call and logs the current stacktrace. Since access to some of the sensors does not require an Android permission, we also manually identified and hooked the functions that give access to non-permission-protected sensor data. Android applications (including the device's browser) cannot directly read the current value of a sensor and are required to register a listener in order to consequently read the captured events. Each sensor can be obtained by calling the `getDefaultSensor()` method of the `android.hardware.SensorManager` class. The listener is declared by specifying the name of the sensor with the `getDefaultSensor()` function, and a `Sensor` instance is created. Finally, the listener is registered by calling the `registerListener()` method. Our module intercepts both of these function calls.

Experimental setup. Among popular browsers for Android, Google Chrome and Mozilla Firefox have better compatibility for HTML5 WebAPIs [52]. Since Chrome relies heavily on Google Play Services for the Android internals, while Firefox more clearly leverages the official and better-documented Android API, we opted for the latter. In our experiments we use Mozilla Firefox (v.59.0.1) as our browser on three Android Google Nexus 5X and a OnePlus One device, all running AOSP 7.1.2. We controlled the devices using custom scripts and the Android Debug Bridge. We first evaluated the effectiveness of our methodology by creating a dummy website that executes all possible mobile HTML5 WebAPIs. Since browsers require a valid certificate in order to call certain APIs (i.e., they are only served over HTTPS) we used a self-signed certificate. Before conducting our actual large-scale study, we confirmed that our approach can successfully intercept and monitor access to the devices' sensors. Our system also simulates brief user interaction through random gestures (swipes and taps) with websites so as to elicit functionality from websites expecting some user activity. Gestures are issued for approximately 30 seconds on average for each website, while

Table 2. Number of domains using mobile WebAPI calls.

WebAPI	#Domains	WebAPI	#Domains
Device orientation	2,199	Ambient light sensor	152
Geolocation	1,688	Proximity sensor	142
Device motion	1,360	Vibration	84
Screen orientation change	645	Media capture	12
Total 6,282			

an extra module monitors for potential redirections to different domains (e.g., due to clicking on an ad) which are rolled back so the original website can continue to be processed.

4 DATA COLLECTION AND ANALYSIS

In this section we present our findings from our large scale study on the use of mobile-specific WebAPI calls in the wild. Our crawling list included the 200K most popular websites according to Alexa, as returned on 03/24/2018, to be processed by our crawling infrastructure comprising of four Android smartphones. Our system was unable to access or complete the crawling process for 16,199 (8%) of the domains in our list (e.g., 503-timeout, 502-Bad Gateway, or DNS errors), and omitted 230 domains flagged as malicious by the Google SafeBrowsing API. Our crawling experiments took place between 03/24/2018-09/03/2018 and 11/11/2018-11/22/2018, from US-based IP addresses.

In Table 2 we can see the prevalence of the mobile-specific WebAPI calls logged by our system among the 183,571 domains processed by our crawling infrastructure. We logged 5,313 (2.89%) websites using at least one of the targeted APIs, while 807 request access to sensor data using more than one of the API calls. The most prevalently accessed data is from the acceleration and orientation sensors which do not require the user's permission, as well as geolocation data which requires permission in major browsers. While the Geolocation API can also return information for desktop computers (using "information about nearby wireless access points and the IP address" [5]), we consider it mobile-specific due to smartphones' integrated GPS receivers which provide real-time location information. While geolocating users based on landline IP addresses is considerably accurate [122], that is not the case for mobile IP addresses [26, 119]. It is important to note that the Media capture and Geolocation APIs should explicitly request permissions from the user; while this is enforced in major browsers, it is not always the case with other browsers (e.g., for Geolocation [67]). For the remaining WebAPI calls, users will be unaware that such information is being retrieved by the website even for major browsers, which occurs in 4,582 (2.49%) of the websites we processed.

As can be seen in Figure 3 the use of mobile-specific WebAPI calls is not uniform across our dataset. Indeed, the highest concentration is found in the top 5K websites with 250 domains, which is more than double the 122 domains in the last chunk 195K-200K. Moreover, all the chunks above the 150 domain-threshold are found within the top 60K. Overall, most chunks contain between 100 and 150 domains requesting access to mobile-specific API calls. Domains also access more *permission-free* WebAPIs (gray bars) independently of their rank, indicating the importance of this sensor data. As discussed later on, this type of information can be used for a plethora of attacks, and users should have the ability to explicitly grant permission for them.

As Figure 4 (left) shows the majority of websites issue request access to a single sensor through the WebAPI, while 15.1% of the domains we processed target at least two different types of sensor data. As shown in Table 1, only accessing the Motion sensor can lead to six different attacks, while

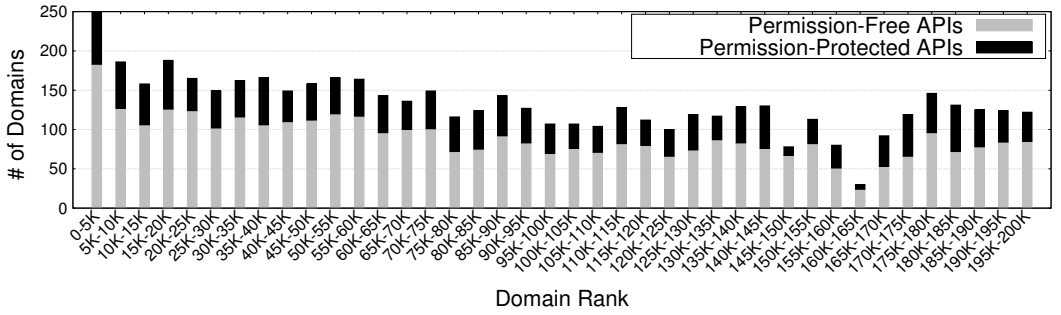

Fig. 3. Number of domains using mobile-specific WebAPI calls. Gray bars indicate domains that only use APIs that do not require a permission, while black bars indicate domains also accessing permission-protected APIs.

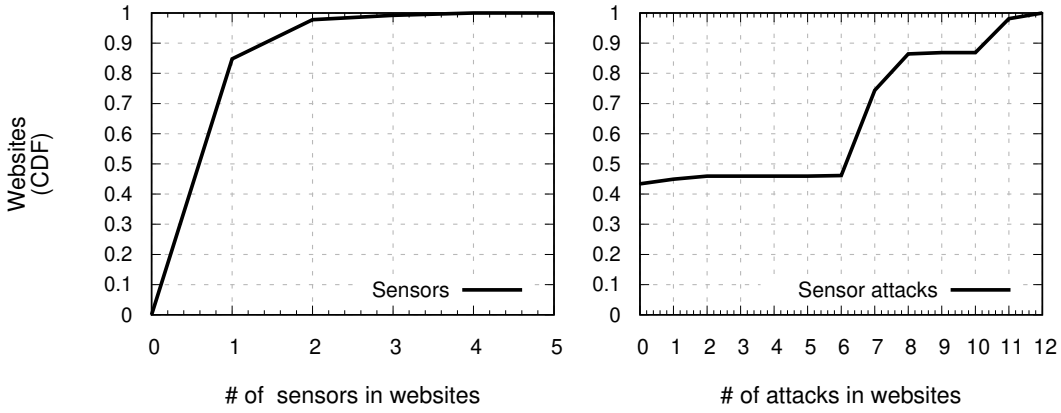

Fig. 4. CDF of websites requesting access to sensors (left) and the number of attacks that are feasible with the data they collect (right).

a combination of two sensors (Motion and Orientation) leads to eight attacks. Furthermore, as can be seen in Figure 4 (right) 56.6% of the domains that issue mobile-specific WebAPI calls are able to perform at least one attack.

Sensor-based attacks. Next, we continue our analysis of the dataset collected by our system by framing it within our taxonomy based on representative prior work. It is important to note that in our analysis we do not take into account or argue for (or against) the *plausibility* of the attacks presented in previous studies. Instead, our goal is to measure the potential risk that mobile users face due to web browsing by identifying websites that request access to specific sensor data and could *potentially* misuse them in an invasive or malicious manner.

Table 3 breaks down the number of domains for each attack. We observe that the most common attacks across websites that access WebAPIs are touchscreen input (55.07%), device/sensor fingerprinting and trait, mood or demographic inference (54.07%), location tracking and mode of transportation (53.85%) and speech recognition (41.39%). As can be seen in Table 3, the most commonly feasible attack enabled by collected mobile sensor data is the inference of the user's touch input. This would allow a malicious domain to exfiltrate extremely sensitive information, such as the user's credit card number, her login credentials, or even private chat messages. The attack

Table 3. Breakdown of sensor based attacks, the number of domains capable of deploying them and the percentage of webpages capable of performing the specific attack.

ID	Mobile Sensor-based Attack	#Domains	Percentage
1	Mode of transportation	2,861	53.85%
2	Body movement or activity	720	13.55%
3	Location tracking	2,861	53.85%
4	Acoustic emanation side-channels	1,372	25.82%
5	Speech recognition	2,199	41.39%
6	Touchscreen input	2,926	55.07%
7	Screen content inference	152	2.86%
8	Inferring user's age	1,360	25.60%
9	Inferring user's sex	2,199	41.39%
10	Inferring user's fingerprints	12	0.23%
11	Inferring user's gait	2,861	53.85%
12	Inferring user's mood	1,372	25.82%
13	Device/sensor fingerprinting	2,873	54.07%
14	Covert channels	96	1.81%

surface grows considerably due to third-party scripts accessing WebAPIs (see Section 4.2), since they create a hidden covert channel and can exfiltrate sensitive data even if the user is browsing a legitimate webpage. We argue that any information gained from sensors poses a risk for users and an access control policy should be enforced, either through some form of run-time permissions [6] or using a mechanism similar to GDPR [3] where users are informed and have to explicitly give their consent.

4.1 In-Depth Analysis and Case Studies

Here we continue our analysis and present a series of case studies. We first classify every domain that accesses at least one WebAPI call using McAfee's real time database [79]. In Table 4 we provide the top 20 categories (sorted in descending order) based on the average number of access requests for sensor data across all the websites of each category. In Appendix A we provide a full list of the classification performed in Table 8. The first column denotes the classification label, while the second, third and fourth columns denote the number of domains, the aggregated number of sensors accessed and the aggregated number of feasible attacks for each category respectively. The last column shows the average access requests for sensors across websites that access at least one mobile sensor. We observe that domains that have a higher request access sensor rate fall into 10 major categories. Domains that fall into these categories typically show a lot of advertisements as well as retargeted ads [29], since users with that kind of browsing history appear to be heavily targeted by advertisers [112]. Based on the plethora of techniques that can be used for the device/sensor fingerprinting attack (see Table 1), we believe that mobile sensors can be used as another channel for tracking users even across sessions. We also observe that the three categories with the highest aggregate number of accessed sensors and attacks are Business, Online Shopping and Entertainment; these categories typically generate more ad revenue than other categories [12]. Since significant effort and deliberate design dictate the rules of digital advertising, the fact that these categories have the highest aggregate numbers of accessed sensors is not coincidental. Indeed ads can influence how we perceive our surroundings, which is highly applicable in beauty products [118]. For instance, a prevalent concept spread by the media relates to how people perceive beauty and attractiveness [27].

Table 4. Domain classification for the top 20 categories sorted in descending order based on the average request access for sensors across websites accessing at least one mobile sensor.

Label	# Domains	# Sensors	# Attacks	% Sensors/Total Domains
Business	662	743	2246	13.98%
Online Shopping	427	539	2316	10.14%
Marketing/Merchandising	417	459	1380	8.64%
Entertainment	348	439	2118	8.26%
Travel	322	392	1492	7.38%
General News	327	385	1640	7.25%
Education/Reference	293	321	1380	6.04%
Internet Services	301	318	1169	5.99%
Finance/Banking	239	307	955	5.78%
Fashion/Beauty	169	249	1218	4.69%
Blogs/Wiki	201	236	1272	4.44%
Public Information	201	226	388	4.25%
Software/Hardware	200	214	569	4.03%
Pornography	136	196	983	3.69%
Potential Illegal Software	117	181	990	3.41%
Health	128	170	480	3.20%
Games	115	143	865	2.69%
Restaurants	126	139	192	2.62%
Sports	113	135	685	2.54%
Real Estate	114	122	252	2.30%

Moreover Barford et al. [28] showed that users with a 'Beauty & Fitness' profile are highly targeted by shopping-related ads.

Figure 5 and Figure 6 depict the most frequently accessed sensors and the most frequently feasible attacks for each domain category. In Figure 5 we observe that the majority of the categories will more often access three specific sensors: the motion, orientation, and device location sensor. We emphasize that the motion and the orientation sensors do not require any permission from the operating system and when used alone lead to 9 and 8 different attacks respectively. Moreover when these three sensors are combined they can result in 12 different attacks, including inferring the touchscreen input, device/sensor fingerprinting, and location tracking. Figure 6 shows that these three are the most frequent attacks across domain categories, along with the inference of demographic information. Interestingly, every category in our dataset, including those that the McAfee service could not classify (i.e., NotAvailable), can infer the user's input across all categories. We argue that the information gained from this attack is extremely sensitive and can lead to more severe attacks.

Domain/Category popularity. To explore whether there is a correlation between the popularity of a given domain category and the number of sensors these domains tend to access, we calculate Pearson's correlation coefficient. Figure 7 shows Pearson's correlation coefficient between the Alexa rank for every website in every domain category with the number of sensors accessed and with the number of attacks that are feasible with the data they collect. A score of 1 denotes perfect linear correlation between two variables, 0 denotes no correlation, and -1 shows total negative correlation. A positive correlation indicates that both variables increase or decrease together, while negative correlation indicates that as one variable increases, so the other decreases, and vice versa. As can be seen in Figure 7 the domain categories that have a positive value indicate that as the

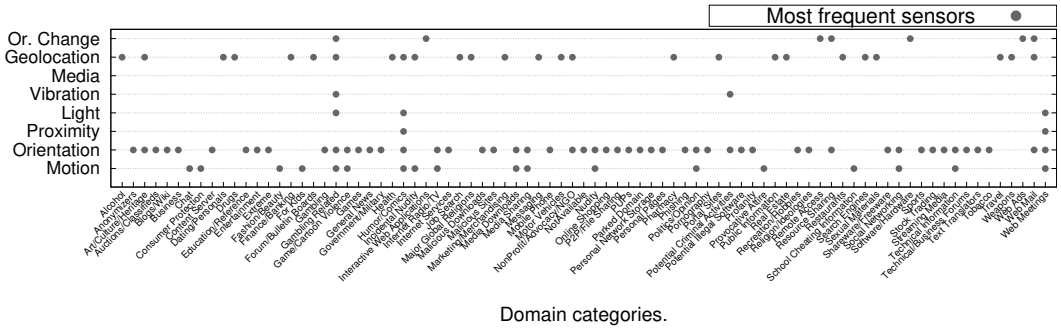

Fig. 5. Most frequent accessed sensors for each domain category.

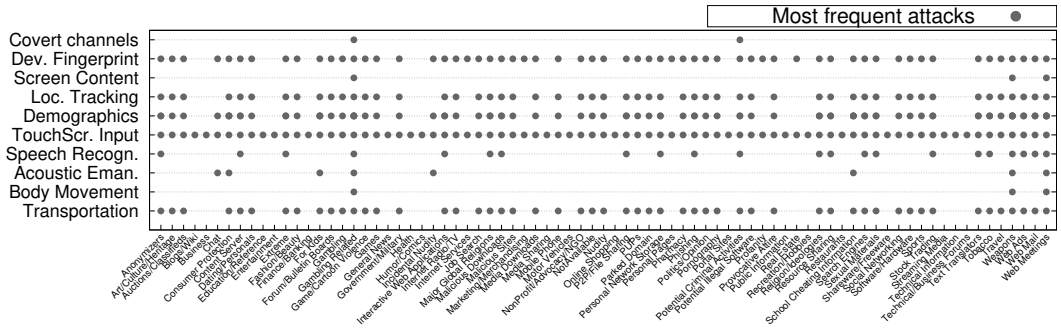

Fig. 6. Most frequent (feasible) attacks for each domain category.

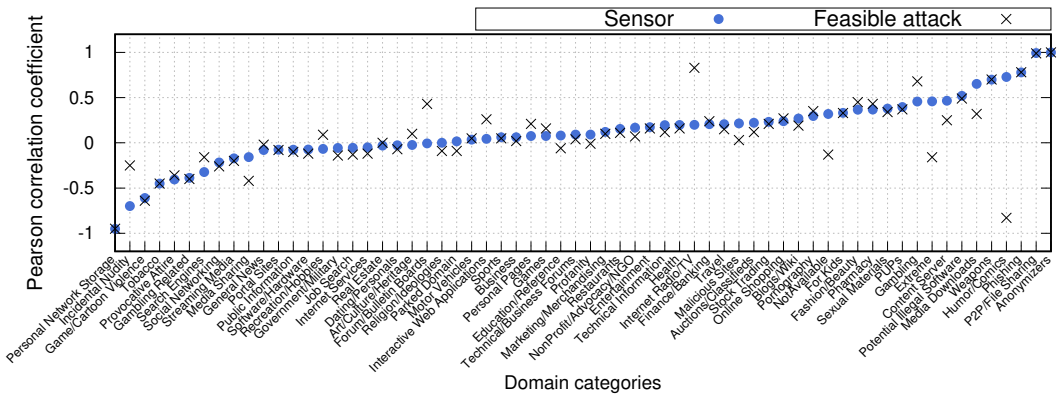

Fig. 7. Pearson's coefficient between the Alexa rank for every website in every domain category, correlated to the number of sensors accessed and the number of attacks that are feasible with the data they collect.

Alexa ranking is dropping so does the number of sensors accessed by websites in this category. The opposite is also true – more popular websites are more likely to access more mobile sensors. This is consistent across different popular categories such as Online Shopping, Finance/Banking, Entertainment, Pornography and Malicious Sites. Websites in these categories have a smaller Alexa ranking, which means that they are more popular and, as we discuss later on, these categories

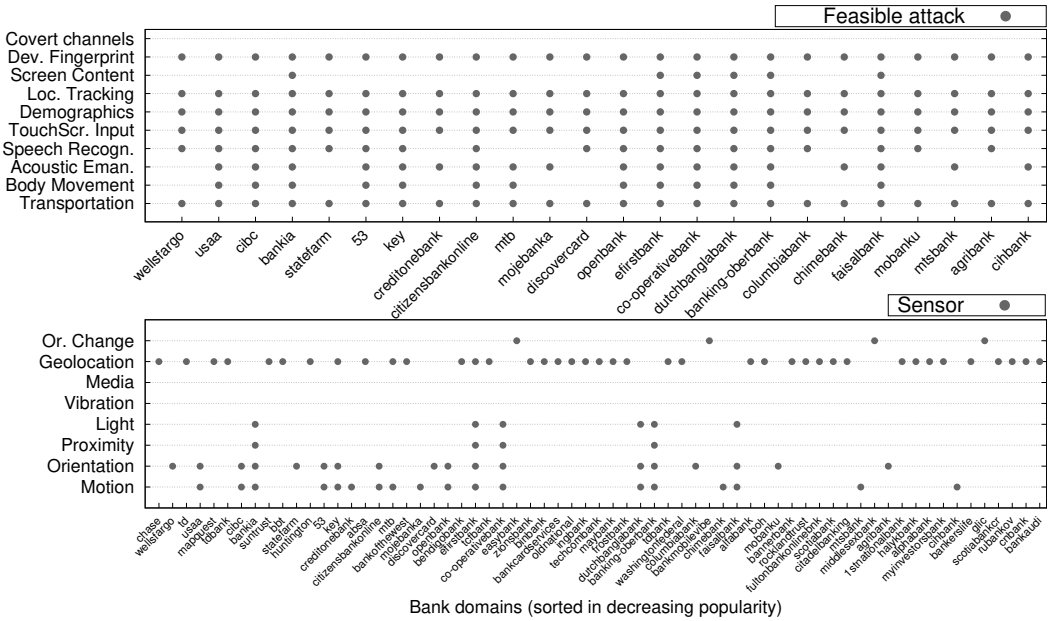

Fig. 8. Breakdown of sensors accessed and corresponding attacks that could be deployed by banking domains.

request a higher average access rate for sensors compared to other categories. Since the number of sensors accessed affects the number of feasible attacks, we deduce that *popular websites and domain categories are prone to access more mobile-specific sensors and such information enables a variety of techniques that can be used for a plethora of attacks*. We also observe that for some categories the r value is zero, indicating that for these categories there is no correlation between the number of sensors accessed and their popularity. For websites with a negative value we observe that as one variable increases (Alexa ranking), so the other decreases (sensors accessed). This is potentially due to these categories containing very few websites as shown in Table 8 in Appendix A.

Banking sites. While device fingerprinting allows third parties to track users across the Web [93], fingerprints can be used as an additional factor for authentication [15]. As such, banking websites are well-suited for deploying such a security mechanism [94] due to the significant implications of compromised accounts. As details of such practices are not typically disclosed, we further explore the prevalence of sensor-based information access across e-banking domains. We compiled a list of bank domains using online resources [1, 7] and cross-referenced it with our dataset.

As can be seen in Figure 8, we identified 65 banking domains that request access to data from at least one mobile sensor. Overall, banking domains request access for 1.38 sensors on average, which is higher than the average of 1.17 in other domains, indicating that banking websites are more likely to leverage the HTML5 WebAPI for accessing sensor data. We find that 24 of the bank domains obtain access to the sensor data necessary to conduct at least one of the attacks included in our taxonomy. Interestingly, all of those banks collect the sensor data leveraged in prior work for device fingerprinting, while 40 banks request access to the user's geolocation which can also be used for enhancing the authentication process [15]. Furthermore, we also find that efirstbank.com actually requests access to more sensors than any other domain in our entire dataset. Overall, while accessing sensor data could be motivated by enhancing the authentication process, this practice raises privacy concerns as argued by privacy advocates [83].

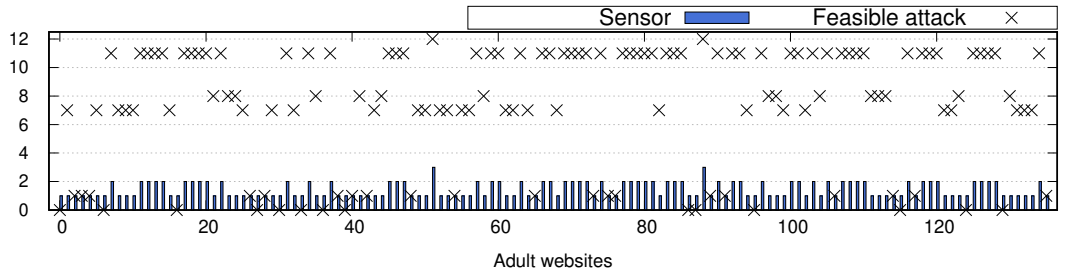

Fig. 9. Breakdown of sensors accessed and corresponding attacks that could be deployed by domains with adult content.

Adult Content. Figure 9 shows the number of sensors accessed by websites that are classified by McAfee as "pornography". We found that 136 domains access at least one mobile sensor. While the majority of them accesses one or two mobile sensors, which can result in 11 different attacks, in aggregate these domains access six out of the eight available sensors (orientation, position, motion, orientation_change, light, vibration). Interestingly we found that 73 domains are capable of inferring the user's age and mood and 87 are capable of inferring the user's sex. Such information can be of great value for this specific category of websites since it can be used to recommend additional content, increasing the duration of users' sessions while showing them targeted advertisements (which leads to increased revenue). The information gained from mobile sensors should not be ignored within this context, especially since third-party analytics and advertising services have the ability to track users across and outside the adult web [121].

Malicious domains. Even though our system checked Google's SafeBrowsing API before visiting a domain, it is possible that visited domains could be flagged as malicious later on, or by different blacklists. As such, we submitted all the domains that issued WebAPI requests to VirusTotal. Figure 10 presents the websites flagged as malicious (sorted by their rank), the number of accessed sensors per website and feasible attacks. Out of those, 149 domains were flagged by one AV engine and 17 domains were flagged by two. We can see that higher ranking malicious domains are more likely to access more sensors which results in a higher number of feasible attacks. We found 11 websites being flagged by at least 3 AV engines. The label on top of the bars in Figure 10 (bottom) represents the number of AV engines identifying these domains as malicious or suspicious. Finally, we found two websites,[1] namely *goggle.com* and *yotube.com*, that are flagged by eight AV engines as malicious. Apart from likely examples of typosquatting [85, 123], these websites requested access to sensor data that could be used to perform one and eleven different attacks respectively.

Country code top-level domain. To get a better understanding of the target audience of the domains that access mobile sensors we plot the websites that access at least one mobile specific WebAPI based on the country code top-level domain. Figure 11 and Figure 12 show the aggregate number of sensors accessed and the aggregate number of feasible attacks. We observe that domains with a code top-level domain from Ukraine, Russia, China, Italy, Canada, Germany, India and Brazil tend to access more sensors than domains from other countries and also have a higher number of aggregate attacks. For example, websites from Ukraine, Russia and China have the ability to perform 909, 795 and 423 attacks respectively. Finally, we observe that some of the countries with the highest aggregate number of feasible attacks are among those that spend more money on digital

[1] The VirusTotal community also confirms that these websites were recently used for malicious purposes, as discussed here [9] and here [10].

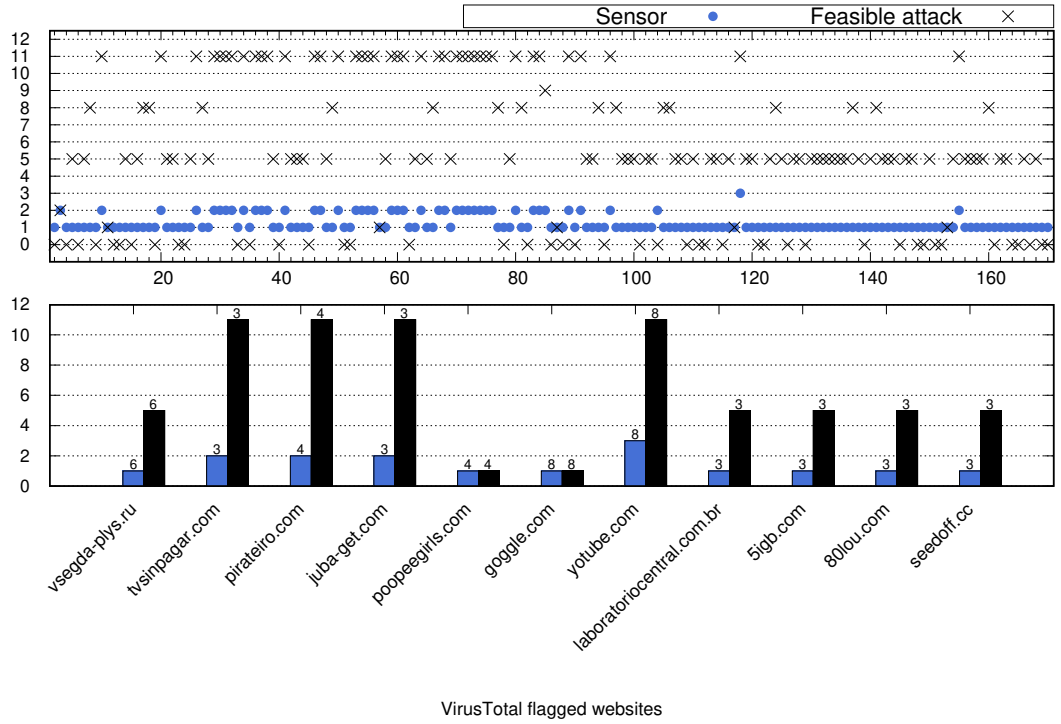

Fig. 10. Number of accessed sensors and feasible attacks for websites flagged as malicious (top). Websites that were flagged by at least three AV engines (bottom) – the number on the bars shows how many AV engines flagged the domain.

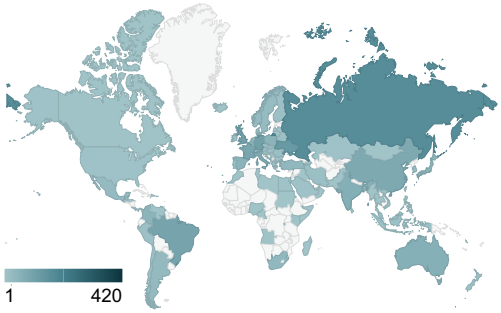

Fig. 11. Aggregate number of sensors accessed by websites, classified by their country code top-level domain.

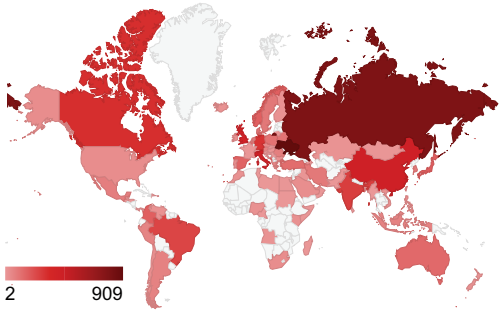

Fig. 12. Aggregate number of feasible attacks for websites, classified by their country code top-level domain.

advertising according to reports (e.g., [13, 14]), indicating that data captured from mobile sensors (e.g., demographics) could potentially be used for enhancing the efficiency of digital advertising. While analyzing how mobile sensor data is leveraged by the online ad ecosystem is out of the scope of this work, we consider an in-depth exploration of this phenomenon an interesting future direction.

4.2 WebAPI Request Origin

Next we explore where the WebAPI requests originate from. Apart from exploring whether the request is first party (i.e., issued from a script hosted on the domain being processed) or third-party (i.e., issued from an external source but inside the original domain's page), our system also logs whether the request originates from an iframe. It is important to note that different browsers implement different policies regarding which sensors can be accessed for these three different types of origin. As such, we present statistics for all the websites that requested access, even if those requests were blocked by Firefox during out experiments.

Iframes. Our system collects all the calls executed by every element of a website, including iframes. In every log we record the source domain name of the element that is accessing sensor information. By comparing the URL of the address bar and the URL in the logfiles, we can identify whether WebAPIs are accessed by the DOM or by an iframe. Our analysis shows that 991 websites out of 5,313 contain iframes that use WebAPIs to access mobile specific information. We analyzed all iframes from our experiments and found that specific iframes are found in different websites. The two most frequent domains injected inside iframes exist in 389 webpages (or 39.3% of pages with iframes collecting data) and are related to online media players.

External sources. Among the websites that issue API calls for mobile specific information we found 40 scripts from external domains (either as a third-party scripts or inside an iframe) that collect data from 2461 websites 46.3%. We manually analyzed these scripts and found that they offer services for media-players and advertisements and they collect information about the orientation and motion of the device. In Table 5 we list the domains that appear in more than 50 websites and collect data from sensors. The first column is the origin of the script being executed. The second and third column show how many websites and iframes host this script while the fourth column is the domain of the page inside the iframe. Given that these third-party domains are used in 35.89% of websites that access sensor data, we classified them based on the type of service they provide using *Cyren*[2]. The last two columns show which sensors the script accessed and their corresponding attacks. We observe that most of these domains call the motion and orientation WebAPIs which enable a plethora of attacks. Moreover, domains classified as search engines and ad-networks gain access to characteristics that can track users across the web.

From Table 5 we can see that the domain *api.b2c.com* enables 12 different attacks. After investigating this domain[3] through VirusTotal [8] we found that scripts served from this domain and Android apps that communicate with it are classified as intrusive adware and even malware by some antivirus vendors. Another domain, *c.adsco.re*, is flagged as malware by Cyren, even though it is not considered malicious by the Google SafeBrowsing API. We manually analyzed the content of the script that retrieves the data and found that apart from retrieving information about the Motion and the Orientation sensors it also exhibits behavior which is a strong indicator of device fingerprinting, such as creating and manipulating canvas elements [87] and reading different Navigator, Screen, Storage and Window properties. Interestingly the *adsco.re* domain states that it is used for traffic validation by Adscore, a bot detection service. In total, these two domains which are considered malicious by certain security lists, were found on 5.4% of all the sensor-accessing domains logged by our system, which again raises concerns regarding browser policies that allow third party domains to access sensor data without explicit user permission.

Android internals. Our crawling system allows an end-to-end analysis of sensor data access. Apart from providing high call-detection fidelity, since we can match requests logged by our injected JavaScript to actions at the operating system level, it also revealed sub-optimal browser behavior.

[2]https://www.cyren.com/security-center/url-category-check
[3]https://www.virustotal.com/#/domain/api.b2c.com

Table 5. Third-party scripts accessing mobile-specific WebAPI calls.

Script origin	#Sites	#iframes	iframe domain	Classification	Sensors	AttackID (see Table 3)
f.vimeocdn.com	275	275	player.vimeo.com	Streaming	O	1, 2, 3, 5, 6, 9, 11, 13
fast.wistia.com	467	3	fast.wistia.com	Technology	OC	-
fast.wistia.net	125	115	fast.wistia.net		OC	-
c.adsco.re	211	-	-	Malware	M,O	1 - 6, 8, 9, 11, 12, 13
g.alicdn.com	170	65	wanwang.aliyun.com m.aliyun.com	Shopping General	O,G	1, 2, 3, 5, 6, 9, 11, 13
aeu.alicdn.com	127	83	mbest.aliexpress.com		O	1, 2, 3, 5, 6, 9, 11, 13
api.b2c.com	76	-	-	General	M,O,P,L	1 - 9, 11, 12, 13
cdn.admixer.net	169	-	-	Ads	M	1 - 4, 6, 8, 11, 12, 13
static.yieldmo.com	107	-	-		O	1, 2, 3, 5, 6, 9, 11, 13
secure-ds.serving-sys.com	51	35	googleads.g.doubleclick.net tpc.googlesyndacation.com	Ads	M	1 - 4 , 6, 8, 11, 12, 13
dlswbr.baidu.com	77	69	pos.baidu.com	Search Engine	M	1 - 4, 6, 8, 11, 12, 13
client.perimeterx.net	73	-	-	Technology	M	1 - 4, 6, 8, 11, 12, 13

M: motion, G: geolocation, P: proximity, O: orientation, L: light, OC: orientation_change

We found that while Firefox prevents iframes from accessing sensor data, in practice Firefox simply "omits" returning the sensor data instead of blocking (i.e., ignoring) the actual request. Specifically, Firefox allows iframes to create event listeners, which then trigger the necessary WebAPI calls which then trigger the corresponding Android-level processing and permission checks for obtaining the sensor data; the data is then returned to the browser but not provided to the iframe.

Android WebView is based on the Chromium project and allows mobile apps to access and display web content. WebView usage is extremely widespread, found in 85% of the apps in the official Google Play Store in 2015 [88]. Due to its prevalence across Android apps, we also tested two popular WebView-based browsers, namely UC, and WebView (info.android1.webview), along with Facebook and Messenger, and found that they allow iframes to obtain data about motion and orientation. As such, even if users use Firefox or Chrome for web browsing, which currently block iframes from accessing any sensor data, opening a website within such popular apps that use WebView would expose them to attacks.

4.3 Transience of Web Measurements

Scheitle et al. [106] recently found that there is significant fluctuation in the websites contained in ranking lists used by academic studies, with Alexa being the most volatile list. As a result, similar measurement experiments that use an Alexa list from a different date could result in a significantly different view of the web ecosystem. To quantify and frame this effect within the dataset we have collected, we compare to the recently released dataset[4] by Das et al. [39] which was part of their concurrent study on mobile sensor fingerprinting. While their collection set up was different (they used a modified version of OpenWPM as opposed to actual mobile devices) they also logged mobile sensor APIs used by popular websites. When comparing the domains that accessed mobile-specific WebAPI calls during our experiments to those in their dataset, we find only 403 overlapping domains – 7.9% of our detected websites. However, our system detected WebAPI calls in 2,252 domains that are in their two US-based datasets but with no calls logged during their experiments. Given that both of our experiments were conducted at similar times, including some overlap in May 2018, and used Alexa's list (our version is from 03/24/2018 while their version is from 05/12/2018), this is a surprising result. As such, the two datasets together

[4]https://databank.illinois.edu/datasets/IDB-9213932#

Table 6. Domains exhibiting differences in the WebAPI calls reported by our system and Das et al. [39].

Website	Our Dataset	[39]	Website	Our Dataset	[39]
allrecipes.com	O	L, M, O, P	britishairways.com	OC, O	M, O
99designs.com	M, OC	M	payback.de	M, P	P
gilt.com	M	M, O	newsela.com	O, P	P
joomshaper.com	O	M, O	udnfunlife.com	O	M, O
beefree.io	O	M, O	ceros.com	O	M, O
360cities.net	M, OC	M	99designs.co.uk	M, OC	M
skyscnr.com	M	O	zerator.com	O	M, O

provide a more extensive coverage of the websites that use WebAPIs to access mobile sensors in the wild (we provide a detailed comparison to their study in Section 6).

Another important dimension that needs to be considered is that the modern web is highly dynamic and websites often introduce new functionality or may even remove existing functionality. To further explore how a view of the web can change through time, we compare the actual WebAPI calls reported for those 403 overlapping domains. While we find that for the vast majority (91.8%) of domains both datasets report the same calls across the two datasets, there are differences for 33 websites. In more detail, for those domains our system logged a total of 74 WebAPI calls, while the datasets from [39] contain 62 calls. This difference is partially due to that study targeting a *subset* of the calls that our study explores. However, in Table 6 we include the remaining domains where the two datasets report different sensor data being requested, which correspond to $\sim 3.47\%$ of the domains detected by both systems. While that number is not very large, it is non-negligible and highlights the dynamic and ever-evolving nature of the web.

A notable example is *allrecipes.com* which in our experiments only obtains the orientation data, while Das et al. reported that it accessed four different sensors. To further investigate this issue, we processed *allrecipes.com* again (10/31/2018) and verified our original findings. Motivated by prior work [74] that leveraged the Internet Archive for obtaining a retrospective view of web tracking, we processed one stored snapshot for each day of the crawling period of that specific dataset as reported by the authors (5/17/2018 - 5/21/2018) using a US-based IP address as well. Again we only identified requests for the device's orientation. Subsequently, we identified the third-party JavaScript file (originating from api.b2c.com) that issued those requests in their dataset, and obtained a snapshot of it from the Internet Archive from 05/15/2018. After de-obfuscating it we verified that it indeed issues those requests.

4.4 Analysis Summary

In our analysis we provide a comprehensive exploration of the mobile sensors that websites access through the use of mobile HTML5 WebAPI calls, and analyze how this data can be used by websites in order to exfiltrate personal information about the user. To that end we have created a taxonomy of sensor-based attacks from prior studies and we present an analysis by framing our collected data within that taxonomy. Our analysis shows that attacks that were previously limited to mobile apps can now migrate to the mobile web, and access to these sensors can lead to a plethora of different attacks such as capturing the user's input, identifying personal information and interests, as well as tracking the user across the web. We subsequently perform an in-depth analysis by classifying these websites into different categories based on the content they provide and analyze our dataset from multiple viewpoints. We find that popular websites and popular categories tend to access more mobile sensors, consequently leading to the feasibility of more sensor-based attacks. Furthermore,

third-party scripts embedded inside webpages are also able to capture sensor information, thus creating a larger attack surface. Compared to similar studies [39], our study focuses on every mobile sensor-based WebAPI call and a direct comparison of the results shows that combining and comparing these datasets highlights the transience of the web ecosystem in practice. To shed more light on this phenomenon and to further facilitate research on the security and privacy risks that users face due to mobile sensor data being accessible to websites, we have made our dataset publicly available. Based on our results we argue that any information gained from sensors poses a risk for users and more effective access control policies should be enforced.

5 DISCUSSION

In this section we provide guidelines for establishing policies and defining the functionality that should be supported and standardized across browsers to better protect users.

Even though the research community has proposed access control mechanisms that regulate which application can access mobile sensors [25] or even provide apps with fake values [11], we believe that these approaches are not sufficient. Specifically, third-parties are able to circumvent such mechanisms either by being embedded in the application's source code or in certain cases by being part of the webpage in the form of third-party JavaScript. For instance, research has shown that browsers that rely on WebView do not enforce correct policies for the HTML5 Geolocation API [68]. Moreover, any approach that relies on completely disabling JavaScript or blocking all scripts will inadvertently lead to poor usability and user experience. To identify such limitations and shortcomings, we performed an empirical analysis of the sensor access control options currently offered by major mobile browsers by examining the security options they provide through their user interface for different use cases. We also experimentally inferred the access control policies in place for different scenarios that pose additional threats to users (e.g., the ability of third-part scripts to access sensor data). Table 7, summarizes our findings. Based on browsers' existing designs, and the limitations we have identified we argue that there is dire need for a standardized, fine-grained universal mechanism that allows users to control access to all types of mobile sensor data. Based on the severe implications of information being gained by these sensors, we provide guidelines and propose a concrete list of access control strategies for different scenarios, that should be adopted across browsers:

- **Universal revocation.** The browser should provide user's with the ability to universally decline access to *all* mobile sensors, regardless of being protected by a specific Android permission. Currently, as shown in Table 7, for the motion sensors (accelerometer and gyroscope) only Google Chrome and Brave (which is based on Chromium) have implemented a feature allowing users to navigate to the browser's site settings under the advanced options in order to disable access to them. Unfortunately, even for these browsers users can not control how access is granted to other mobile sensors (e.g., Ambient Light). Moreover, users do not have the option to enable access for a specific sensor or a specific website (i.e., a whitelist-based approach where a user can explicitly allow a specific website to always access a given sensor).
- **Explicit permission requests - Permissions list.** The browser should enforce its own run-time permission system for *all* mobile sensors. This feature is already being used for the GPS sensor; whenever a website requires access to the device's location, the user can grant or deny the request. This feature not only informs users about the functionality of the website but also allows for a fine-grained access control mechanism where users have control over their data. Since explicitly asking the user for permission to access the sensor at each request can lead to poor usability and user experience, the browser should maintain a list of domains that the user has granted or denied access to, similar to Android's dangerous permissions

for apps. This will allow users to revoke access to a specific sensor for a specific domain at any time. A permission list has been implemented in Google Chrome for regulating access to different elements of a specific website (Sound, Mic, Location, etc.) and can be found in the `Site settings` under the permission category. Unfortunately even in the latest version of the mobile Google Chrome (version 79), the permission list does not support or include mobile sensors. Interestingly the desktop version of Google Chrome includes motion sensors in the permission list found under the same `Site settings` category. Since motion sensors exists mostly in mobile devices it is likely that this feature may soon be implemented in the mobile version of Google Chrome. Due to the severity of the attacks enabled by mobile sensors we argue that whenever a website requests access to any of the available mobile sensors the user should be informed with an explicit permission request.

- **Origin differentiation.** As shown by our experiments, the most commonly feasible sensor-based attack is the inference of the user's touch input which allows a malicious domain to exfiltrate extremely sensitive information, such as the user's credit card number. The attack surface grows considerably since third-party scripts can create a hidden covert channel and exfiltrate sensitive data even if the user is browsing a legitimate webpage. Even though only one of the browsers tested (UC) allows iframes to gain access to motion sensors, we found that they *all* allow third-party scripts to obtain data about the device's motion and orientation. We believe that the browser should inform the user whether the sensor request originates from the first party domain or from a script hosted on a third-party domain. This information about the origin will enable users to make better decisions about whether to grant permission or not. Indeed, prior work [46] highlighted the need for providing Android API permissions based on the origin of the request – a similar idea can be applied at the application layer for differentiating access control policies based on a script's origin. Moreover, studies have shown that users are more likely to deny a permission request when a detailed description of the data that will be accessed is given [48]. Understanding the purpose of why and how sensitive resources are used can have a major impact on their feelings and trust decisions [75].
- **Private browsing.** Private browsing was created as a privacy feature where the browser creates a temporary session isolated from the browser's main session and user data. In the current web ecosystem where ad platforms and trackers collect an abundance of user information that is added to user profiles so as to improve recommendations, private browsing modes allow users to browse the web without exposing the common identifiers (i.e., cookies) that are sent by browsers during normal operation. Unfortunately, data from smartphone sensors can be used to accurately fingerprint a mobile device and, as an extension, the user. Therefore we argue that browsers should deny access to all mobile sensors in private browsing mode by default, unless users explicitly change their settings to allow that. Through empirical analysis of the most popular mobile browsers, (including Google Chrome, Mozilla Firefox and Microsoft Edge) we have found that they *all* allow access to sensors while in private browsing mode. As shown in Table 7, even privacy-oriented browsers such as Brave and DuckDuckGo neglect to block sensor access in private browsing mode.

6 RELATED WORK

HTML5 WebAPI. The introduction of WebAPI has standardized many functions and features providing greater support for developers, enriching websites and web apps and improving the user experience [101]. Snyder et al. [110] presented a study on the use of HTML5 functions in a small set (10K) of popular websites. The functionality provided by HTML5 allows users to experience multimedia content without the hassle and, more importantly, the vulnerabilities introduced by external plugins or proprietary software, such as the Adobe Flash plugin which was progressively

Table 7. Access control currently enforced for *motion sensors* in popular mobile browsers. The "Universal revocation", "Per-site revocation" and "Origin differentiation" columns indicate whether the browser supports this feature. Columns marked with an asterisk (*) indicate whether the browser allows access to motion sensors in those scenarios.

Browser	Version	Universal revocation	Per-site revocation	Origin differ/tion	Third-party scripts*	iframes*	Private browsing*
Chrome	79.0.39	✓	✗	✗	✓	✗	✓
Firefox	68.4.2	✗	✗	✗	✓	✗	✓
Edge	44.11.2	✗	✗	✗	✓	✗	✓
Brave	1.5.3	✓	✗	✗	✓	✗	✓
Opera Mini	46.0.22	✗	✗	✗	✓	✗	✓
UC Browser	v12.12.6	✗	✗	✗	✓	✓	✓
DuckDuckGo	5.41.0	✗	✗	✗	✓	✗	✓
Dolphin	v12.1.5	✗	✗	✗	✓	✗	✓

abandoned and substituted with HTML5 media elements [76]. At the same time, smartphone browsing has become very popular in the past years, with devices facilitating browsing even in screens that are comparatively small. This transition was possible with the introduction of multi-touch gestures and the performance improvement of mobile devices and the mobile network [50].

In an independent and concurrent recent study Das et al [39] presented a study on web scripts accessing mobile sensors. While their study also targets WebAPI calls for mobile sensors, our work presents significant differences. In regards to the actual datasets, our study is on a considerably larger set of domains while also having little overlap due to the fluctuation of the Alexa list [106]. Moreover, their system detects a subset of the mobile-specific WebAPI calls handled by our system, and their study focuses only on sensor-based fingerprinting. As such, their study offers a limited examination of the risks that users face; our study frames our findings within our attack taxonomy and provides a more comprehensive evaluation of the feasibility of a wide range of sensor-based attacks. Furthermore, our crawling infrastructure uses actual mobile devices and provides a unique end-to-end view of data requests and access from the application layer down to the operating-system Android internals. Their crawlers rely on a modified version of OpenWPM running on desktop machines, which could be detected by evasive websites, e.g., through canvas and emoji elements [72], that subsequently alter their behavior. Nonetheless, we believe that their study provides an important and complimentary view of the mobile web ecosystem, and the combination of our findings and public datasets will be useful resources for the research community.

Browser fingerprinting has gathered a lot of attention in recent years and the research community has extensively studied the techniques that make it possible [49]. Eckersley [47] introduced the Panopticlick project and explored browser fingerprinting in depth. With the growing usage of smartphones, traditional desktop fingerprinting techniques [120] (e.g., canvas, screen and graphics fingerprinting, etc.), are becoming less effective as some information is being standardized in many mobile browsers [65].

On the other hand, the development of new mobile-specific HTML5 WebAPIs offered new avenues for trackers to exploit other types of data that were not present in desktops. As previous work [16, 17, 32, 40–43, 45, 51, 60, 65, 66, 82, 90, 102, 103, 131, 133] has shown, the huge amount of input collected by smartphones sensors resulted in new opportunities for device fingerprinting. A notable case is by Olejnik et al. [95], that explores how the Battery Status API yields information about the maximum capacity of the battery and the discharge time, which can be used to effectively

track users across the web. This attracted a lot of attention which resulted in Firefox discontinuing its support of the Battery API.

Modern smartphones contain a wide range of sensors that collect information about the current state of the device and the environment surrounding it. Websites and applications have access to these sensors with most of them (e.g., gyroscope, accelerometer, proximity etc.) not explicitly requiring the permission of the user. Bai et al. [25] provided a solution by instrumenting the original .apk and enforcing a policy controller. Papadopoulos et al. [98] proposed an anti-blocking mechanism for PII. Other countermeasures that address information leakage include the use of browser extensions. Starov et al. [114] developed PrivacyMeter, an extension that calculates on-the-fly a privacy score for each visited website. Merzdovnik et al. [81] study the effectiveness of the most popular extensions in identifying and blocking trackers from third-party libraries. Snyder et al. [111] developed an extension that is able to manipulate web-pages and block specific API calls. Even though extensions are widely supported by most common browsers for desktop platforms they are not widely available for mobile devices. Google Chrome for Android does not support extensions, nor does Firefox in iOS.

7 CONCLUSION

We presented a comprehensive evaluation of the threats that mobile users face when browsing the Web, due to capabilities offered by modern browsers. Specifically, we conducted the largest and most extensive study to date on the use of mobile-specific WebAPI calls in the wild. Our study was conducted using a novel crawling infrastructure built on top of actual smartphone devices, allowing us to trace data requests and access from the application layer down to the Android operating system internals. Our findings demonstrate that WebAPI capabilities are actively being used by websites for accessing mobile sensors. To provide the appropriate context that highlights the true threat posed by this practice, we created a taxonomy of sensor-based attacks compiled from a wide range of attacks demonstrated in prior work. Our subsequent in-depth analysis correlated the sensor data currently being accessed by websites and the data-requirements of prior attacks, leading to several alarming findings. Apart from the fact that the vast majority of the websites that leverage mobile-specific WebAPI calls can carry out privacy-invasive attacks, we also found that 5.4% of those websites included third-party scripts that accessed sensor data and were hosted on domains flagged as malware by security services. We believe that our findings support the need for more stringent policies for websites attempting to access sensor data, allowing users to explicitly declare preferences and set their own privacy policy.

ACKNOWLEDGMENTS

This project has received funding from Horizon 2020 under grant agreements No 786890 (THREAT-ARREST), No 830927 (CONCORDIA), and No 833683 (CyberSANE), the DARPA ASED Program and AFRL under contract FA8650-18-C-7880 and NSF under contract CNS-1934597. This paper reflects only the view of the authors and the funding bodies are not responsible for any use that may be made of the information it contains.

REFERENCES

[1] [n.d.]. Alexa - Top 50 Banks and Institutions. https://www.alexa.com/topsites/category/Business/Financial_Services/Banking_Services/Banks_and_Institutions.
[2] [n.d.]. Axplorer - demystifying the Android Application Framework. http://axplorer.org/.
[3] [n.d.]. The EU General Data Protection Regulation. https://eugdpr.org.
[4] [n.d.]. MDN Web Docs - Magnetometer. https://developer.mozilla.org/en-US/docs/Web/API/Magnetometer/Magnetometer.

[5] [n.d.]. Mozilla Support - Does Firefox share my location with websites? https://support.mozilla.org/en-US/kb/does-firefox-share-my-location-websites.

[6] [n.d.]. Request prompts for dangerous permissions. https://developer.android.com/guide/topics/permissions/overview#dangerous-permission-prompt.

[7] [n.d.]. US Banks on the Internet. http://www.thecommunitybanker.com/bank_links/.

[8] [n.d.]. VirusTotal: Analyze suspicious files and URLs to detect types of malware. https://www.virustotal.com.

[9] [n.d.]. VirusTotal: goggle.com. https://tinyurl.com/VTgoggle-com.

[10] [n.d.]. VirusTotal: yotube.com. https://tinyurl.com/VTyotube-com.

[11] 2016. The ultimate, yet easy to use, privacy manager for Android. https://github.com/M66B/XPrivacy.

[12] 2019. IAB FY 2018 Podcast Ad Revenue Study. https://www.iab.com/wp-content/uploads/2019/06/Full-Year-2018-IAB-Podcast-Ad-Rev-Study_6.03.19_vFinal.pdf.

[13] 2019. What's Shaping the Digital Ad Market. https://www.emarketer.com/content/global-digital-ad-spending-2019.

[14] 2019-07-22. Year-over-year change of advertising expenditure in selected countries from 2016 to 2018. https://www.statista.com/statistics/276805/global-advertising-market-forecast/.

[15] Furkan Alaca and Paul C van Oorschot. 2016. Device fingerprinting for augmenting web authentication: classification and analysis of methods. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*. ACM, 289–301.

[16] Irene Amerini, Rudy Becarelli, Roberto Caldelli, Alessio Melani, and Moreno Niccolai. 2017. Smartphone fingerprinting combining features of on-board sensors. *IEEE Transactions on Information Forensics and Security* 12, 10 (2017), 2457–2466.

[17] Irene Amerini, Paolo Bestagini, Luca Bondi, Roberto Caldelli, Matteo Casini, and Stefano Tubaro. 2016. Robust smartphone fingerprint by mixing device sensors features for mobile strong authentication. *Electronic Imaging* 2016, 8 (2016), 1–8.

[18] S Abhishek Anand and Nitesh Saxena. 2018. Speechless: Analyzing the Threat to Speech Privacy from Smartphone Motion Sensors. In *2018 IEEE Symposium on Security and Privacy (SP). Vol. 00*. 116–133.

[19] Andrei Popescu. 2018. Geolocation API. https://www.w3.org/TR/geolocation-API/. Accessed: 2018-07-13.

[20] Anssi Kostiainen. 2018. Ambient light sensor API. https://www.w3.org/TR/ambient-light/. Accessed: 2018-07-13.

[21] Anssi Kostiainen. 2018. Vibration API. https://www.w3.org/TR/vibration/. Accessed: 2018-07-13.

[22] Anssi Kostiainen, Alexander Shalamov. 2018. Accelerometer. https://www.w3.org/TR/accelerometer/. Accessed: 2018-07-13.

[23] Anssi Kostiainen, Rijubrata Bhaumik. 2018. Proximity sensor API. https://www.w3.org/TR/proximity/. Accessed: 2018-07-13.

[24] Adam J Aviv, Benjamin Sapp, Matt Blaze, and Jonathan M Smith. 2012. Practicality of accelerometer side channels on smartphones. In *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 41–50.

[25] Xiaolong Bai, Jie Yin, and Yu-Ping Wang. 2017. Sensor Guardian: prevent privacy inference on Android sensors. *EURASIP Journal on Information Security* 2017, 1 (2017), 10.

[26] Mahesh Balakrishnan, Iqbal Mohomed, and Venugopalan Ramasubramanian. 2009. Where's That Phone?: Geolocating IP Addresses on 3G Networks. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement (IMC '09)*.

[27] Anna M. Bardone-Cone and Kamila M. Cass. 2006. Investigating the impact of pro-anorexia websites: a pilot study. *European Eating Disorders Review* 14, 4 (2006), 256–262. https://doi.org/10.1002/erv.714 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/erv.714

[28] Paul Barford, Igor Canadi, Darja Krushevskaja, Qiang Ma, and S. Muthukrishnan. 2014. Adscape: Harvesting and Analyzing Online Display Ads. In *Proceedings of the 23rd International Conference on World Wide Web* (Seoul, Korea) *(WWW '14)*. ACM, New York, NY, USA, 597–608. https://doi.org/10.1145/2566486.2567992

[29] Muhammad Ahmad Bashir, Sajjad Arshad, William Robertson, and Christo Wilson. 2016. Tracing Information Flows Between Ad Exchanges Using Retargeted Ads. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 481–496. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/bashir

[30] Ben Alman. 2018. Monkey-patch (hook) functions for debugging and stuff. https://github.com/cowboy/javascript-hooker. Accessed: 2018-04-23.

[31] Sebastian Biedermann, Stefan Katzenbeisser, and Jakub Szefer. 2015. Hard drive side-channel attacks using smartphone magnetic field sensors. In *International Conference on Financial Cryptography and Data Security*. Springer, 489–496.

[32] Hristo Bojinov, Yan Michalevsky, Gabi Nakibly, and Dan Boneh. 2014. Mobile device identification via sensor fingerprinting. *arXiv preprint arXiv:1408.1416* (2014).

[33] Armir Bujari, Bogdan Licar, and Claudio E Palazzi. 2012. Movement pattern recognition through smartphone's accelerometer. In *2012 IEEE Consumer Communications and Networking Conference (CCNC)*. IEEE, 502–506.

[34] Liang Cai and Hao Chen. 2011. TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion. *HotSec* 11 (2011), 9–9.
[35] Liang Cai and Hao Chen. 2012. On the practicality of motion based keystroke inference attack. In *International Conference on Trust and Trustworthy Computing*. Springer, 273–290.
[36] Supriyo Chakraborty, Wentao Ouyang, and Mani Srivastava. 2017. LightSpy: Optical eavesdropping on displays using light sensors on mobile devices. In *Big Data (Big Data), 2017 IEEE International Conference on*. IEEE, 2980–2989.
[37] Cortesi, Aldo and Hils, Mayimilian and Kriechbaumer, Thomas. [n.d.]. mitmproxy. https://mitmproxy.org. v. 3.0.3.
[38] Daniel C. Burnett, Adam Bergkvist, Cullen Jennings, Anant Narayanan, Bernard Aboba. 2018. Media capture API. https://www.w3.org/TR/mediacapture-streams/. Accessed: 2018-07-13.
[39] Anupam Das, Gunes Acar, Nikita Borisov, and Amogh Pradeep. 2018. The Web's Sixth Sense: A Study of Scripts Accessing Smartphone Sensors. In *Proceedings of ACM CCS, October 2018*.
[40] Anupam Das, Nikita Borisov, and Matthew Caesar. 2014. Do you hear what i hear?: Fingerprinting smart devices through embedded acoustic components. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 441–452.
[41] Anupam Das, Nikita Borisov, and Matthew Caesar. 2016. Tracking Mobile Web Users Through Motion Sensors: Attacks and Defenses.. In *NDSS*.
[42] Anupam Das, Nikita Borisov, and Edward Chou. 2018. Every Move You Make: Exploring Practical Issues in Smartphone Motion Sensor Fingerprinting and Countermeasures. *Proceedings on Privacy Enhancing Technologies* 2018, 1 (2018), 88–108.
[43] Erhan Davarci, Betul Soysal, Imran Erguler, Sabri Orhun Aydin, Onur Dincer, and Emin Anarim. 2017. Age group detection using smartphone motion sensors. In *Signal Processing Conference (EUSIPCO), 2017 25th European*. IEEE, 2201–2205.
[44] Luke Deshotels. 2014. Inaudible Sound as a Covert Channel in Mobile Devices.. In *WOOT*.
[45] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. 2014. AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable.. In *NDSS*.
[46] Michalis Diamantaris, Elias P. Papadopoulos, Evangelos P. Markatos, Sotiris Ioannidis, and Jason Polakis. 2019. REAPER: Real-time App Analysis for Augmenting the Android Permission System. In *9th ACM Conference on Data and Application Security and Privacy, CODASPY '19*. ACM.
[47] Peter Eckersley. 2010. How unique is your web browser?. In *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 1–18.
[48] Nicole Eling, Siegfried Rasthofer, Max Kolhagen, Eric Bodden, and Peter Buxmann. 2016. Investigating Users' Reaction to Fine-Grained Data Requests: A Market Experiment. In *HICSS '16*.
[49] Steven Englehardt and Arvind Narayanan. 2016. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1388–1401.
[50] Fazal-e-Amin. 2015. Characterization of web browser usage on smartphones. *Computers in human behavior* 51 (Oct. 2015), 896–902. https://doi.org/10.1016/j.chb.2014.10.054
[51] Tobias Fiebig, Jan Krissler, and Ronny Hänsch. 2014. Security Impact of High Resolution Smartphone Cameras.. In *WOOT*.
[52] Maximiliano Firtman. 2018. Mobile HTML5 Compatibility on Mobile Devices. http://mobilehtml5.org/. Accessed: 2018-04-22.
[53] Matteo Gadaleta and Michele Rossi. 2018. Idnet: Smartphone-based gait recognition with convolutional neural networks. *Pattern Recognition* 74 (2018), 25–37.
[54] Daniel Genkin, Mihir Pattani, Roei Schuster, and Eran Tromer. 2019. Synesthesia: Detecting Screen Content via Remote Acoustic Side Channels. In *2019 IEEE Symposium on Security and Privacy (SP)*.
[55] Daniel Genkin, Adi Shamir, and Eran Tromer. 2014. RSA key extraction via low-bandwidth acoustic cryptanalysis. In *International cryptology conference*. Springer, 444–461.
[56] ghostwords. 2018. Browser fingerprinting protection for everybody. https://github.com/ghostwords/chameleon. Accessed: 2018-06-09.
[57] Global Stats. 2018. Mobile and tablet internet usage exceeds desktop for first time worldwide. http://gs.statcounter.com/press/mobile-and-tablet-internet-usage-exceeds-desktop-for-first-time-worldwide. Accessed: 2018-04-19.
[58] Glenn Greenwald. 2014. *No place to hide: Edward Snowden, the NSA, and the US surveillance state*. Macmillan.
[59] Daniel Gruss, David Bidner, and Stefan Mangard. 2015. Practical memory deduplication attacks in sandboxed javascript. In *European Symposium on Research in Computer Security*. Springer, 108–122.
[60] Jun Han, Emmanuel Owusu, Le T Nguyen, Adrian Perrig, and Joy Zhang. 2012. Accomplice: Location inference using accelerometers on smartphones. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*. IEEE, 1–9.

[61] Samuli Hemminki, Petteri Nurmi, and Sasu Tarkoma. 2013. Accelerometer-based transportation mode detection on smartphones. In *Proceedings of the 11th ACM conference on embedded networked sensor systems*. ACM, 13.

[62] Duncan Hodges and Oliver Buckley. 2018. Reconstructing what you said: Text Inference using Smartphone Motion. *IEEE Transactions on Mobile Computing* (2018).

[63] Shaohan Hu, Lu Su, Shen Li, Shiguang Wang, Chenji Pan, Siyu Gu, Md Tanvir Al Amin, Hengchang Liu, Suman Nath, Romit Roy Choudhury, and Tarek F. Abdelzaher. 2015. Experiences with eNav: A Low-power Vehicular Navigation System. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Osaka, Japan) *(UbiComp '15)*. ACM, New York, NY, USA, 433–444. https://doi.org/10.1145/2750858.2804287

[64] Jingyu Hua, Zhenyu Shen, and Sheng Zhong. 2017. We can track you if you take the metro: Tracking metro riders using accelerometers on smartphones. *IEEE Transactions on Information Forensics and Security* 12, 2 (2017), 286–297.

[65] Thomas Hupperich, Davide Maiorca, Marc Kührer, Thorsten Holz, and Giorgio Giacinto. 2015. On the robustness of mobile device fingerprinting: Can mobile users escape modern web-tracking mechanisms?. In *Proceedings of the 31st Annual Computer Security Applications Conference*. ACM, 191–200.

[66] Felix Juefei-Xu, Chandrasekhar Bhagavatula, Aaron Jaech, Unni Prasad, and Marios Savvides. 2012. Gait-id on the move: Pace independent human identification using cell phone accelerometer dynamics. In *Biometrics: Theory, Applications and Systems (BTAS), 2012 IEEE Fifth International Conference on*. IEEE, 8–15.

[67] Hyungsub Kim, Sangho Lee, and Jong Kim. 2014. Exploring and mitigating privacy threats of HTML5 geolocation API. In *Proceedings of the 30th Annual Computer Security Applications Conference*. ACM, 306–315.

[68] Hyungsub Kim, Sangho Lee, and Jong Kim. 2014. Exploring and Mitigating Privacy Threats of HTML5 Geolocation API. In *Proceedings of the 30th Annual Computer Security Applications Conference* (New Orleans, Louisiana, USA) *(ACSAC '14)*. ACM, New York, NY, USA, 306–315. https://doi.org/10.1145/2664243.2664247

[69] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. 2011. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter* 12, 2 (2011), 74–82.

[70] Nicholas D Lane, Petko Georgiev, and Lorena Qendro. 2015. DeepEar: robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 283–294.

[71] Pierre Laperdrix. 2017. *Browser Fingerprinting: Exploring Device Diversity to Augment Authentification and Build Client-Side Countermeasures*. Ph.D. Dissertation. Rennes, INSA.

[72] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. 2016. Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. In *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 878–894.

[73] Adam Lella. 2018. U.S. Smartphone Penetration Surpassed 80 Percent in 2016. https://www.comscore.com/Insights/Blog/US-Smartphone-Penetration-Surpassed-80-Percent-in-2016. Accessed: 2018-04-18.

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

[75] Jialiu Lin, Shahriyar Amini, Jason I. Hong, Norman Sadeh, Janne Lindqvist, and Joy Zhang. 2012. Expectation and Purpose: Understanding Users' Mental Models of Mobile App Privacy Through Crowdsourcing. In *UbiComp '12*.

[76] Yogesh Maheshwari and Y Raghu Reddy. 2017. A study on Migrating Flash files to HTML5/JavaScript. In *Proceedings of the 10th Innovations in Software Engineering Conference*. ACM, 112–116.

[77] Claudio Marforio, Hubert Ritzdorf, Aurélien Francillon, and Srdjan Capkun. 2012. Analysis of the communication between colluding applications on modern smartphones. In *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 51–60.

[78] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. 2011. (sp) iPhone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 551–562.

[79] McAfee. 2019. Customer URL Ticketing System, Check Single URL. https://trustedsource.org/. Accessed: 2019-04-22.

[80] Maryam Mehrnezhad, Ehsan Toreini, Siamak F Shahandashti, and Feng Hao. 2018. Stealing PINs via mobile sensors: actual risk versus user perception. *International Journal of Information Security* 17, 3 (2018), 291–313.

[81] Georg Merzdovnik, Markus Huber, Damjan Buhov, Nick Nikiforakis, Sebastian Neuner, Martin Schmiedecker, and Edgar Weippl. 2017. Block me if you can: A large-scale study of tracker-blocking tools. In *Security and Privacy (EuroS&P), 2017 IEEE European Symposium on*. IEEE, 319–333.

[82] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. 2014. Gyrophone: Recognizing Speech from Gyroscope Signals.. In *USENIX Security Symposium*. 1053–1067.

[83] Elinor Mills. [n.d.]. Device identification in online banking is privacy threat, expert says. https://www.cnet.com/news/device-identification-in-online-banking-is-privacy-threat-expert-says/.

[84] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. 2012. Tapprints: your finger taps have fingerprints. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACm, 323–336.

[85] Tyler Moore and Benjamin Edelman. 2010. Measuring the perpetrators and funders of typosquatting. In *International Conference on Financial Cryptography and Data Security*. Springer, 175–191.
[86] Mounir Lamouri, Marcos Cáceres. 2018. Screen orientation API. https://www.w3.org/TR/screen-orientation/. Accessed: 2018-07-13.
[87] Keaton Mowery and Hovav Shacham. 2012. Pixel Perfect: Fingerprinting Canvas in HTML5. In *Proceedings of W2SP 2012*.
[88] Patrick Mutchler, Adam Doupé, John Mitchell, Chris Kruegel, and Giovanni Vigna. 2015. A large-scale study of mobile web app security. In *Proceedings of the Mobile Security Technologies Workshop (MoST)*.
[89] Sashank Narain, Amirali Sanatinia, and Guevara Noubir. 2014. Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning. In *Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks*. ACM, 201–212.
[90] Sashank Narain, Triet D Vo-Huu, Kenneth Block, and Guevara Noubir. 2016. Inferring user routes and locations using zero-permission mobile sensors. In *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 397–413.
[91] Sarfraz Nawaz and Cecilia Mascolo. 2014. Mining Users' Significant Driving Routes with Low-power Sensors. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems* (Memphis, Tennessee) *(SenSys '14)*. ACM, New York, NY, USA, 236–250. https://doi.org/10.1145/2668332.2668348
[92] Khuong An Nguyen, Raja Naeem Akram, Konstantinos Markantonakis, Zhiyuan Luo, and Chris Watkins. 2019. Location Tracking Using Smartphone Accelerometer and Magnetometer Traces. In *Proceedings of the 14th International Conference on Availability, Reliability and Security* (Canterbury, CA, United Kingdom) *(ARES '19)*. ACM, New York, NY, USA, Article 96, 9 pages. https://doi.org/10.1145/3339252.3340518
[93] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. 2013. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Security and privacy (SP), 2013 IEEE symposium on*. IEEE, 541–555.
[94] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. 2014. On the workings and current practices of web-based device fingerprinting. *IEEE Security & Privacy* 12, 3 (2014), 28–36.
[95] Łukasz Olejnik, Gunes Acar, Claude Castelluccia, and Claudia Diaz. 2015. The leaking battery. In *Data Privacy Management, and Security Assurance*. Springer, 254–263.
[96] Lukasz Olejnik, Steven Englehardt, and Arvind Narayanan. 2017. Battery Status Not Included: Assessing Privacy in Web Standards. In *3rd International Workshop on Privacy Engineering (IWPE'17). San Jose, United States*.
[97] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. 2012. ACCessory: password inference using accelerometers on smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*. ACM, 9.
[98] Elias P Papadopoulos, Michalis Diamantaris, Panagiotis Papadopoulos, Thanasis Petsas, Sotiris Ioannidis, and Evangelos P Markatos. 2017. The long-standing privacy debate: Mobile websites vs mobile apps. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee.
[99] Dan Ping, Xin Sun, and Bing Mao. 2015. TextLogger: Inferring Longer Inputs on Touch Screen Using Motion Sensors. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks* (New York, New York) *(WiSec '15)*. ACM, New York, NY, USA, Article 24, 12 pages. https://doi.org/10.1145/2766498.2766511
[100] Rahul Raguram, Andrew M. White, Dibyendusekhar Goswami, Fabian Monrose, and Jan-Michael Frahm. 2011. iSpy: Automatic Reconstruction of Typed Input from Compromising Reflections. In *Proceedings of the 18th ACM Conference on Computer and Communications Security* (Chicago, Illinois, USA) *(CCS '11)*. ACM, New York, NY, USA, 527–536. https://doi.org/10.1145/2046707.2046769
[101] Ashis Kumar Ratha, Shibani Sahu, and Priya Meher. 2018. HTML5 in Web Development: A New Approach. (2018).
[102] Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. 2010. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks (TOSN)* 6, 2 (2010), 13.
[103] Yanzhi Ren, Yingying Chen, Mooi Choo Chuah, and Jie Yang. 2013. Smartphone based user verification leveraging gait recognition for mobile healthcare systems. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2013 10th Annual IEEE Communications Society Conference on*. IEEE, 149–157.
[104] Rich Tibbett, Tim Volodine, Steve Block, Andrei Popescu. 2018. Device orientation event. https://www.w3.org/TR/orientation-event/. Accessed: 2018-07-13.
[105] rovo89. 2018. Xposed framework. https://repo.xposed.info. Accessed: 2018-06-14.
[106] Quirin Scheitle, Oliver Hohlfeld, Julien Gamba, Jonas Jelten, Torsten Zimmermann, Stephen D. Strowes, and Narseo Vallina-Rodriguez. 2018. A Long Way to the Top: Significance, Structure, and Stability of Internet Top Lists. In *IMC*.
[107] Roman Schlegel, Kehuan Zhang, Xiao-yong Zhou, Mehool Intwala, Apu Kapadia, and XiaoFeng Wang. 2011. Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones.. In *NDSS*, Vol. 11. 17–33.

[108] Chao Shen, Shichao Pei, Zhenyu Yang, and Xiaohong Guan. 2015. Input extraction via motion-sensor behavior analysis on smartphones. *Computers & Security* 53 (2015), 143–155.

[109] Laurent Simon and Ross Anderson. 2013. Pin skimmer: Inferring pins through the camera and microphone. In *Proceedings of the Third ACM workshop on Security and privacy in smartphones & mobile devices*. ACM, 67–78.

[110] Peter Snyder, Lara Ansari, Cynthia Taylor, and Chris Kanich. 2016. Browser feature usage on the modern web. In *Proceedings of the 2016 Internet Measurement Conference*. ACM, 97–110.

[111] Peter Snyder, Cynthia Taylor, and Chris Kanich. 2017. Most Websites Don't Need to Vibrate: A Cost-Benefit Approach to Improving Browser Security. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 179–194.

[112] Konstantinos Solomos, Panagiotis Ilia, Sotiris Ioannidis, and Nicolas Kourtellis. 2019. TALOS: An automated framework for Cross-Device Tracking Detection. In *International Symposium on Research in Attacks, Intrusions, and Defenses*.

[113] Raphael Spreitzer. 2014. Pin skimming: Exploiting the ambient-light sensor in mobile devices. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*. ACM, 51–62.

[114] Oleksii Starov and Nick Nikiforakis. 2018. PrivacyMeter: Designing and Developing a Privacy-Preserving Browser Extension. In *International Symposium on Engineering Secure Software and Systems*. Springer, 77–95.

[115] Greg Sterling. [n.d.]. Mobile Devices Now Driving 56 Percent Of Traffic To Top Sites.

[116] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. 2015. Smart Devices Are Different: Assessing and MitigatingMobile Sensing Heterogeneities for Activity Recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems* (Seoul, South Korea) *(SenSys '15)*. ACM, New York, NY, USA, 127–140. https://doi.org/10.1145/2809695.2809718

[117] Yuan Tian, Ying Chuan Liu, Amar Bhosale, Lin Shung Huang, Patrick Tague, and Collin Jackson. 2014. All your screens are belong to us: Attacks exploiting the html5 screen sharing api. In *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 34–48.

[118] Debra Trampe, Diederik A. Stapel, and Frans W. Siero. 2010. The Self-Activation Effect of Advertisements: Ads Can Affect Whether and How Consumers Think about the Self. *Journal of Consumer Research* 37, 6 (10 2010), 1030–1045. https://doi.org/10.1086/657430 arXiv:http://oup.prod.sis.lan/jcr/article-pdf/37/6/1030/5283859/37-6-1030.pdf

[119] Sipat Triukose, Sebastien Ardon, Anirban Mahanti, and Aaditeshwar Seth. 2012. Geolocating IP Addresses in Cellular Data Networks. In *Proceedings of the 13th International Conference on Passive and Active Measurement (PAM'12)*.

[120] Randika Upathilake, Yingkun Li, and Ashraf Matrawy. 2015. A classification of web browser fingerprinting techniques. In *New Technologies, Mobility and Security (NTMS), 2015 7th International Conference on*. IEEE, 1–5.

[121] Pelayo Vallina, Alvaro Feal, Julien Gamba, Narseo Vallina-Rodriguez, and Antonio Fernandez Anta. 2019. Tales from the Porn: A Comprehensive Privacy Analysis of the Web Porn Ecosystem. In *Proceedings of the 2019 Internet Measurement Conference*.

[122] Yong Wang, Daniel Burgener, Marcel Flores, Aleksandar Kuzmanovic, and Cheng Huang. 2011. Towards Street-Level Client-Independent IP Geolocation.. In *NSDI*, Vol. 11. 27–27.

[123] Yi-Min Wang, Doug Beck, Jeffrey Wang, Chad Verbowski, and Brad Daniels. 2006. Strider Typo-Patrol: Discovery and Analysis of Systematic Typo-Squatting. *SRUTI* 6 (2006), 31–36.

[124] Takuya Watanabe, Mitsuaki Akiyama, and Tatsuya Mori. 2015. RouteDetector: Sensor-based positioning system that exploits spatio-temporal regularity of human mobility. In *9th {USENIX} Workshop on Offensive Technologies ({WOOT} 15)*.

[125] Yuxi Wu, Panya Gupta, Miranda Wei, Yasemin Acar, Sascha Fahl, and Blase Ur. 2018. Your Secrets Are Safe: How Browsers' Explanations Impact Misconceptions About Private Browsing Mode. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 217–226.

[126] Yuanyi Wu, Dongyu Meng, and Hao Chen. 2017. Evaluating private modes in desktop and mobile browsers and their resistance to fingerprinting. In *Communications and Network Security (CNS), 2017 IEEE Conference on*. IEEE, 1–9.

[127] Chenren Xu, Sugang Li, Gang Liu, Yanyong Zhang, Emiliano Miluzzo, Yih-Farn Chen, Jun Li, and Bernhard Firner. 2013. Crowd++: Unsupervised Speaker Count with Smartphones. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Zurich, Switzerland) *(UbiComp '13)*. ACM, New York, NY, USA, 43–52. https://doi.org/10.1145/2493432.2493435

[128] Zhi Xu, Kun Bai, and Sencun Zhu. 2012. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*.

[129] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. 2017. DeepSense: A Unified Deep Learning Framework for Time-Series Mobile Sensing Data Processing. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 351–360. https://doi.org/10.1145/3038912.3052577

[130] Jiexin Zhang, Alastair Beresford, and Ian Sheret. 2019. Sensorid: Sensor calibration fingerprinting for smartphones. In *2019 IEEE Symposium on Security and Privacy (SP)*.

[131] Zhe Zhou, Wenrui Diao, Xiangyu Liu, and Kehuan Zhang. 2014. Acoustic fingerprinting revisited: Generate stable device id stealthily with inaudible sound. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 429–440.

[132] Tong Zhu, Qiang Ma, Shanfeng Zhang, and Yunhao Liu. 2014. Context-free Attacks Using Keyboard Acoustic Emanations. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (Scottsdale, Arizona, USA) *(CCS '14)*. ACM, New York, NY, USA, 453–464. https://doi.org/10.1145/2660267.2660296

[133] John Zulueta, Andrea Piscitello, Mladen Rasic, Rebecca Easter, Pallavi Babu, Scott A Langenecker, Melvin McInnis, Olusola Ajilore, Peter C Nelson, Kelly Ryan, et al. 2018. Predicting Mood Disturbance Severity with Mobile Phone Keystroke Metadata: A BiAffect Digital Phenotyping Study. *Journal of medical Internet research* 20, 7 (2018).

A APPENDIX

Table 8. Domain classification sorted in descending order based on the average request access for sensors across websites accessing at least one mobile sensor.

Label	# Domains	# Sensors	# Attacks	% Sensors/Total Domains
Business	662	743	2246	13.98%
Online Shopping	427	539	2316	10.14%
Marketing/Merchandising	417	459	1380	8.64%
Entertainment	348	439	2118	8.26%
Travel	322	392	1492	7.38%
General News	327	385	1640	7.25%
Education/Reference	293	321	1380	6.04%
Internet Services	301	318	1169	5.99%
Finance/Banking	239	307	955	5.78%
Fashion/Beauty	169	249	1218	4.69%
Blogs/Wiki	201	236	1272	4.44%
Public Information	201	226	388	4.25%
Software/Hardware	200	214	569	4.03%
Pornography	136	196	983	3.69%
Potential Illegal Software	117	181	990	3.41%
Health	128	170	480	3.20%
Games	115	143	865	2.69%
Restaurants	126	139	192	2.62%
Sports	113	135	685	2.54%
Real Estate	114	122	252	2.30%
Motor Vehicles	105	111	252	2.09%
Government/Military	64	96	364	1.81%
Portal Sites	81	88	304	1.66%
Recreation/Hobbies	63	71	300	1.34%
Forum/Bulletin Boards	64	68	240	1.28%
Job Search	64	68	164	1.28%
NonProfit/Advocacy/NGO	46	58	188	1.09%
Streaming Media	33	43	210	0.81%
Technical/Business Forums	34	43	204	0.81%
NotAvailable	35	39	222	0.73%
Auctions/Classifieds	28	31	150	0.58%
PUPs	26	30	176	0.56%
Gambling	15	23	97	0.43%
Parked Domain	19	23	85	0.43%
Pharmacy	19	23	71	0.43%
Search Engines	18	21	46	0.40%
Dating/Personals	17	20	70	0.38%
Media Sharing	13	17	70	0.32%
Interactive Web Applications	15	16	46	0.30%
Religion/Ideologies	13	16	65	0.30%
Provocative Attire	12	15	80	0.28%
Social Networking	13	15	82	0.28%
Art/Culture/Heritage	13	14	55	0.26%
Content Server	12	14	51	0.26%
Stock Trading	11	14	54	0.26%

Continued on next page

Table 8 – continued from previous page

Label	# Domains	# Sensors	# Attacks	% Sensors/Total Domains
Technical Information	12	13	62	0.24%
Personal Pages	10	12	77	0.23%
Internet Radio/TV	8	9	56	0.17%
Sexual Materials	8	9	18	0.17%
Weapons	7	9	12	0.17%
Major Global Religions	8	8	21	0.15%
Mobile Phone	8	8	44	0.15%
Incidental Nudity	6	7	24	0.13%
Malicious Sites	6	7	47	0.13%
Shareware/Freeware	7	7	39	0.13%
Alcohol	6	6	0	0.11%
Gambling Related	4	6	13	0.11%
Game/Cartoon Violence	4	6	37	0.11%
Media Downloads	5	6	27	0.11%
P2P/File Sharing	4	6	36	0.11%
Politics/Opinion	6	6	45	0.11%
Tobacco	5	6	39	0.11%
Chat	5	5	22	0.09%
Humor/Comics	3	5	17	0.09%
Nudity	5	5	30	0.09%
Phishing	3	5	25	0.09%
Profanity	4	5	30	0.09%
School Cheating Information	5	5	40	0.09%
Drugs	4	4	0	0.08%
Extreme	3	4	19	0.08%
For Kids	3	4	27	0.08%
Personal Network Storage	3	4	25	0.08%
Web Ads	4	4	7	0.08%
Web Meetings	1	4	12	0.08%
Anonymizers	2	3	18	0.06%
Web Mail	3	3	7	0.06%
Potential Criminal Activities	2	2	8	0.04%
Resource Sharing	2	2	7	0.04%
Consumer Protection	1	1	8	0.02%
Malicious Downloads	1	1	7	0.02%
Messaging	1	1	0	0.02%
Remote Access	1	1	0	0.02%
Text Translators	1	1	7	0.02%