A Large-scale Study on the Risks of the HTML5 WebAPI for Mobile Sensor-based Attacks

Francesco Marcantoni
Univ. of Illinois at Chicago, USA
fmarca2@uic.edu

Michalis Diamantaris
FORTH, Greece
diamant@ics.forth.gr

Sotiris Ioannidis
FORTH, Greece
sotiris@ics.forth.gr

Jason Polakis
Univ. of Illinois at Chicago, USA
polakis@uic.edu

ABSTRACT

Smartphone sensors can be leveraged by malicious apps for a plethora of different attacks, which can also be deployed by malicious websites through the HTML5 WebAPI. In this paper we provide 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 used in the wild. 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. Our study reveals the extent to which websites are actively leveraging the WebAPI for collecting sensor data, with 2.89% of websites accessing at least one mobile sensor. To provide a comprehensive assessment of the potential 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 could carry out at least one of those attacks. Our findings emphasize the need for a standardized policy across browsers and the ability for users to control what sensor data each website can access.

CCS CONCEPTS

• **Security and privacy → Mobile platform security**.

KEYWORDS

Android; mobile HTML5; sensor attack taxonomy; WebAPI

ACM Reference Format:
Francesco Marcantoni, Michalis Diamantaris, Sotiris Ioannidis, and Jason Polakis. 2019. A Large-scale Study on the Risks of the HTML5 WebAPI for Mobile Sensor-based Attacks. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3308558.3313539

1 INTRODUCTION

Smartphones have become almost ubiquitous, with the volume of Internet traffic from mobile devices surpassing that of desktop computers worldwide [35], while 56% of the traffic to top-sites in the US was from mobile devices [65]. Apart from the obvious usability benefits smartphones offer, they have also introduced a plethora of risks. 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 [70, 71]. However, adversaries can still track users through fingerprints [53], making it possible to identify which device is navigating a given webpage [42]. Prior work has also shown that imperfections in sensors' hardware render them fingerprintable [25]. Websites can access such mobile sensor data through the widely supported HTML5 WebAPI. Thus, 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 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 Android devices and perform an end-to-end analysis of WebAPI requests. Using our crawling infrastructure, we measure the prevalence of mobile-specific WebAPI calls across 183,571 of the most popular websites during March-November 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. To better understand the implications 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 could be potentially carried out by modern websites and conduct an in-depth analysis of our dataset. We argue that with 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 build a novel crawling infrastructure and conduct a high-fidelity, large-scale end-to-end study of websites targeting mobile-specific sensors.
- We provide a taxonomy of previously reported sensor-based attacks and reframe them within the modern mobile ecosystem. Guided by our taxonomy we conduct a qualitative analysis of our collected data and a provide a comprehensive assessment of the threat posed by the mobile WebAPI.
- We publicly release our data: https://www.cs.uic.edu/~webapi.

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313539

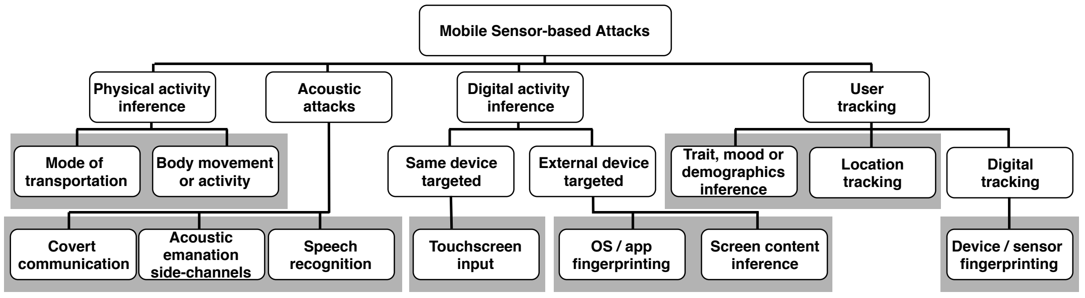

Figure 1: Taxonomy of attacks demonstrated in prior studies that leverage data from mobile sensors.

2 BACKGROUND

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 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.

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 describe the 4 main classes from our taxonomy's first level and refer to some of the presented attacks.

Physical activity inference. Numerous studies [27, 36, 38, 46, 52, 58] 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 [58].

Acoustic attacks. [11, 24, 32, 33, 44, 45, 47, 62] showed that access to Motion, Orientation or the Vibration API can be used to infer users' credit card numbers by listening for specific frequencies [62] or what a user is typing on the keyboard [45], and bypassing dynamic analysis and antivirus through covert channel attacks [44].

Digital activity inference. This class includes a wide range of attacks, with prior work [14, 16, 17, 30, 37, 46, 56, 64, 72] 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., [46, 56]). 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 [17].

User tracking. User tracking has garnered much attention [9, 10, 15, 20–23, 25, 30, 36, 39, 40, 47, 52, 58, 59, 73, 74], and can be conducted in different ways – from coarse-grained location tracking that does not require any user-permission (using the Accelerometer or Gyroscope) [36, 52], to fine-grained device fingerprinting using rich and high-resolution data from smartphone sensors(e.g., [10, 25]). In this category we also include alternative attacks that could track users by inferring demographic information (e.g., age [23], gender [47] and fingerprints [30]), physical traits such as their gait [40, 59], or information about their mental state or mood [74].

Deconstructing sensor attacks. Table 1 lists the attacks 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. For example [58] infers the body movement or activity of a user by accessing the motion and GPS sensors. On the other hand, [52] only requires the orientation sensor, but using the Motion or Magnetometer sensor can further improve the attack.

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.

System architecture. Our system employs a proxy server that intercepts network traffic by using `mitmproxy` [18]. We configured all the Android devices used in our experiments with `mitm`'s certificate in order to intercept both HTTP and HTTPS traffic. 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 Xposed [60] module that (i) detects and hooks requests to sensor-specific Android API calls and (ii) identifies which of these API calls are permission-protected. By intercepting these low-level function calls we are able

Table 1: Sensor based side-channel attacks.

Ref.#	Attack	Motion	Orien/tion	Light	Vibr/tion	Magn/ter	Media (Cam,	Mic)	GPS
[58]	Mode of transportation	◖	-	-	-	-	-	-	◖
[36, 38]		●	-	-	-	-	-	-	-
[52]		○	●	-	-	○	-	-	-
[58]	Movement/activity	◖	-	-	-	-	-	-	◖
[46]		◖	◖	-	-	-	-	-	-
[58]	Location Tracking	◖	-	-	-	-	-	-	◖
[36]		●	-	-	-	-	-	-	-
[52]		○	●	-	-	○	-	-	-
[45]	Acoustic emanation side-channels	●	-	-	-	-	-	-	-
[32, 33]		-	-	-	-	-	-	●	-
[47]	Speech recognition	-	●	-	-	-	-	-	-
[11]		◖	◖	-	-	-	-	-	-
[16, 37, 46, 72]	Touchscreen input	◖	◖	-	-	-	-	-	-
[56]		●	-	-	-	-	-	-	-
[64]		-	-	●	-	-	-	-	-
[30]		-	-	-	-	-	●	-	-
[14]	OS/app fingerprinting	-	-	-	-	●	-	-	-
[17]	Screen inference	-	-	●	-	-	-	-	-
[23, 59, 74]	Physical or demographics	●	-	-	-	-	-	-	-
[47]		-	●	-	-	-	-	-	-
[30]		-	-	-	-	-	●	-	-
[40]		◖	◖	-	-	-	-	-	-
[21, 22, 39]	Device/sensor fingerprinting	◖	◖	-	-	-	-	-	-
[15]		●	-	-	-	-	-	●	-
[10]		●,◖	●	-	-	-	◖	-	-
[25]		●	-	-	-	-	-	-	-
[20, 73]		-	-	-	-	-	-	●	-
[9]		◖	◖	-	-	◖	-	◖	-
[44]	Covert communication side-channels	-	-	-	●	-	-	-	-
[62]		-	-	-	-	-	-	●	-
[24]		◖	-	-	◖	-	-	-	-

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 that enhances the accuracy of the attack with (○). Grey columns denote sensor data that should require explicit user permission according to the W3C.

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 [31].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.

JavaScript Calls Interception. We build our component for hooking JavaScript methods upon the `javascript-hooker` Node.js module [13]. In our experiments we do not overwrite the original function but only need to identify whether a function is called. Thus, whenever a 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 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 [34] and overwrite the `getter` property of each event prototype. 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. 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 [26] module that hooks every Android permission protected API call. 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 apps (including the 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 declaring a listener and specifying its name with the `getDefaultSensor()` function. Then, the listener is registered using the `registerListener()` method. Our module intercepts both of these function calls.

Experimental setup. 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 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 we used a self-signed certificate and confirmed that our approach can successfully intercept and monitor access to the devices' sensors. Our system also simulates brief user interaction through random gestures which are issued for approximately 30 seconds on average for each website, while an extra module monitors for potential redirections to different domains which are rolled back so the original website can continue to be processed.

Table 2: Number of domains using mobile WebAPIs 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			

4 DATA COLLECTION AND ANALYSIS

Our crawling list included the 200K most popular websites according to Alexa, as returned on 03/24/2018. 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" [4]), 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 [68], that is not the case for mobile IP addresses [12, 66]. 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 [41]). For the remaining WebAPI calls, users will be unaware that such information is being retrieved by the website even for major browsers.

As Figure 2 shows the use of mobile-specific WebAPI calls is not uniform across our dataset. The highest concentration is found in the top 5K websites with 250 domains. We observe that domains also access more *permission-free* WebAPIs (gray bars) independently of their rank, indicating the importance of this sensor data. This type of information can be used for a plethora of attacks, and users should have the ability to explicitly grant permission for them.

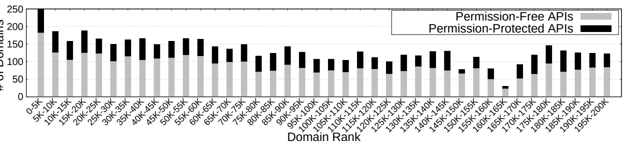

Figure 2: Number of domains using mobile WebAPI calls.

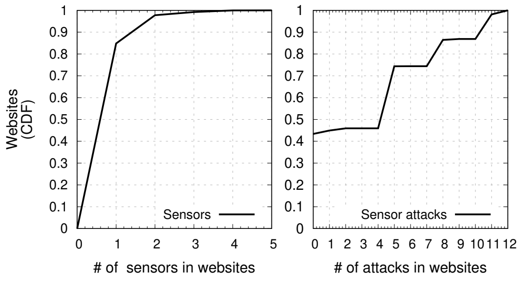

Figure 3: CDF of websites requesting access to sensors (left) and the number of feasible attacks (right).

Table 3: Breakdown of sensor based attacks and the number of domains capable of deploying them.

ID	Mobile Sensor-based Attack	#Domains
1	Mode of transportation	2,861
2	Body movement or activity	720
3	Location tracking	2,861
4	Acoustic emanation side-channels	1,372
5	Speech recognition	2,199
6	Touchscreen input	1,425
7	Screen content inference	152
8	Inferring user's age	1,360
9	Inferring user's sex	2,199
10	Inferring user's fingerprints	12
11	Inferring user's gait	1,360
12	Inferring user's mood	1,360
13	Device/sensor fingerprinting	2,873
14	Covert channels	96

As Figure 3 (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 a combination of two sensors (Motion and Orientation) leads to eight attacks. Furthermore, as can be seen in Figure 3 (right) 56.6% of the domains that issue mobile-specific WebAPI calls are able to perform at least one attack.

Sensor-based attacks. We continue our analysis by framing our dataset 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 device/sensor fingerprinting and trait, mood or demographic inference (54.07%), location tracking and mode of transportation (53.84%), speech recognition (41.38%) and touch-screen input (26.82%). 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 [5] or using a mechanism similar to GDPR [3].

Banking sites. While fingerprinting allows third parties to track users across the Web [53], fingerprints can be used as an additional factor for authentication [8]. As such, banking websites are well-suited for deploying such a security mechanism [54] 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 [1, 6] and cross-referenced it with our dataset. We identified 65 banking domains that request access to at least one mobile sensor. Banking domains request access for 1.38 sensors on average, which is higher than the average of 1.17 in other domains, indicating that they are more likely to leverage the HTML5 WebAPI for accessing sensor data. We find that 24 of the 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 [8]. We 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 [48].

Request origin. Next we explore the origin of WebAPI requests (first or third party) and whether it was included in an iframe. Different browsers implement different policies regarding which sensors can be accessed for these three different types of origin. We present statistics for all the websites that requested access, even if those requests were blocked by Firefox during our 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 4 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. 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*. 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 4 we can see that the domain *api.b2c.com* enables 12 different attacks. After investigating this domain through VirusTotal [7] 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 [50] 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. 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.

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 4 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. Finally, we found two websites, namely *goggle.com* and *yotube.com*, that are flagged by eight AV engines as malicious. Apart from likely examples of typosquatting [49, 69], these websites requested access to sensor data that could be used to perform one and eleven different attacks respectively.

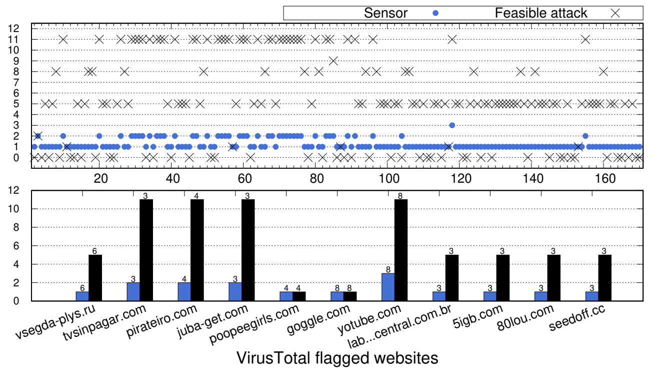

Figure 4: Number of accessed sensors and feasible attacks for websites flagged as malicious (top). Websites that were flagged by at least three AV engines (bottom) – the number on the bars shows how many AV engines flagged the domain.

Table 4: Third-party scripts accessing WebAPI calls.

Script origin	#Sites	iframes	Sensors	AttackID
f.vimeocdn.com	275	275	O	1, 3, 5, 9
fast.wistia.com	467	3	OC	-
fast.wistia.net	125	115	OC	-
c.adsco.re	211	-	M,O	1-6, 8, 9, 11-13
g.alicdn.com	170	65	O,G	1, 3, 5, 9
aeu.alicdn.com	127	83	O	1, 3, 5, 9
api.b2c.com	76	-	M,O,P,L	1-9, 11-13
cdn.admixer.net	169	-	M	1, 3, 4, 6, 8, 11-13
static.yieldmo.com	107	-	O	1, 3, 5, 9
secure-ds.serving-sys.com	51	35	M	1, 3, 4, 6, 8, 11-13
dlswbr.baidu.com	77	69	M	1, 3, 4, 6, 8, 11-13
client.perimeterx.net	73	-	M	1, 3, 4, 6, 8, 11-13

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

WebView usage is extremely widespread [51], so we tested three popular WebView-based browsers, namely Dolphin, UC, and WebView (info.android1.webview), along with Facebook and Messenger, and found that they all allow iframes to obtain motion and orientation data. As such, even if users use Firefox or Chrome for browsing, which currently block iframes from accessing sensor data, clicking a link within such popular apps can expose them to attacks.

Transience of web measurements. Scheitle et al. [61] found 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 by Das et al. [19] 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.

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 [19] contain 62 calls. This difference is partially due to that study targeting a *subset* of the calls that our study explores. However, there are other domains [2] 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.

5 RELATED WORK

The WebAPI has standardized many features providing greater support for developers and improving the user experience [57]. Snyder et al. [63] presented a cost-benefit analysis of the WebAPI using a small set of websites (10K) and focusing on desktop browsing.

In an independent and concurrent recent study Das et al. [19] 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 [61]. Moreover, their system detects a subset of the mobile-specific WebAPI calls handled by our system, and their study focuses on sensor-based fingerprinting thus offering a limited examination of the risks that users face; we frame our findings within our attack taxonomy and provide 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, while their crawlers rely on a modified version of OpenWPM running on desktop machines which could be detected by evasive websites [43].

Browser fingerprinting has gathered a lot of attention and the research community has extensively studied the techniques that make it possible [28, 29]. With the growing usage of smartphones, traditional desktop fingerprinting techniques [67] are becoming less effective as some information is being standardized in many mobile browsers [39]. 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 [9, 10, 15, 20–23, 25, 30, 36, 39, 40, 47, 52, 55, 58, 59, 73, 74] has shown, the huge amount of input collected by smartphones sensors resulted in new opportunities for device fingerprinting.

6 CONCLUSION

We presented a comprehensive evaluation of the threats that mobile users face when browsing the Web, due to capabilities offered by modern browsers, by conducting 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 smartphones. 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. We argue that our findings support the need for more stringent policies for websites attempting to access sensor data.

ACKNOWLEDGMENTS

This project has received funding from Horizon 2020 under grant agreements No 786890 and No 740787, the DARPA ASED Program and AFRL under contract FA8650-18-C-7880. 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.]. Domains exhibiting differences in the WebAPI calls reported by our system and Das et al. https://www.cs.uic.edu/~webapi/.
[3] [n. d.]. The EU General Data Protection Regulation. https://eugdpr.org.
[4] [n. d.]. Mozilla Support - Does Firefox share my location with websites? https://support.mozilla.org/en-US/kb/does-firefox-share-my-location-websites.
[5] [n. d.]. Request prompts for dangerous permissions. https://developer.android.com/guide/topics/permissions/overview#dangerous-permission-prompt.
[6] [n. d.]. US Banks on the Internet. http://www.thecommunitybanker.com/bank_links/.
[7] [n. d.]. VirusTotal: Analyze suspicious files and URLs to detect types of malware. https://www.virustotal.com.
[8] 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.
[9] 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.
[10] 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.
[11] 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.
[12] Mahesh Balakrishnan, Iqbal Mohomed, and Venugopalan Ramasubramanian. [n. d.]. Where's That Phone?: Geolocating IP Addresses on 3G Networks. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement (IMC '09)*.
[13] Ben Alman. [n. d.]. Monkey-patch (hook) functions for debugging and stuff. https://github.com/cowboy/javascript-hooker. Accessed: 2018-04-23.
[14] 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.
[15] Hristo Bojinov, Yan Michalevsky, Gabi Nakibly, and Dan Boneh. 2014. Mobile device identification via sensor fingerprinting. *arXiv preprint arXiv:1408.1416* (2014).
[16] Liang Cai and Hao Chen. 2011. TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion. *HotSec* 11 (2011), 9–9.
[17] 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.
[18] Cortesi, Aldo and Hils, Mayimilian and Kriechbaumer, Thomas. [n. d.]. mitmproxy. https://mitmproxy.org. v. 3.0.3.
[19] 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*.
[20] 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.

[21] Anupam Das, Nikita Borisov, and Matthew Caesar. 2016. Tracking Mobile Web Users Through Motion Sensors: Attacks and Defenses.. In *NDSS*.

[22] 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.

[23] 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.

[24] Luke Deshotels. 2014. Inaudible Sound as a Covert Channel in Mobile Devices.. In *WOOT*.

[25] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. 2014. AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable.. In *NDSS*.

[26] 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.

[27] Kostas Drakonakis, Panagiotis Ilia, Sotiris Ioannidis, and Jason Polakis. 2019. Please Forget Where I Was Last Summer: The Privacy Risks of Public Location (Meta)Data. In *26th Annual Network and Distributed System Security Symposium (NDSS '19)*.

[28] Peter Eckersley. 2010. How unique is your web browser?. In *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 1–18.

[29] 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.

[30] Tobias Fiebig, Jan Krissler, and Ronny Hänsch. 2014. Security Impact of High Resolution Smartphone Cameras.. In *WOOT*.

[31] Maximiliano Firtman. [n. d.]. Mobile HTML5 Compatibility on Mobile Devices. http://mobilehtml5.org/. Accessed: 2018-04-22.

[32] Daniel Genkin, Mihir Pattani, Roei Schuster, and Eran Tromer. 2018. Synesthesia: Detecting Screen Content via Remote Acoustic Side Channels. In *International cryptology conference*.

[33] Daniel Genkin, Adi Shamir, and Eran Tromer. 2014. RSA key extraction via low-bandwidth acoustic cryptanalysis. In *International cryptology conference*. Springer, 444–461.

[34] ghostwords. [n. d.]. Browser fingerprinting protection for everybody. https://github.com/ghostwords/chameleon. Accessed: 2018-06-09.

[35] Global Stats. [n. d.]. 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.

[36] 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.

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

[38] 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.

[39] 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.

[40] 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.

[41] 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.

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

[43] 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.

[44] 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.

[45] 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.

[46] 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.

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

[48] 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/.

[49] 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.

[50] Keaton Mowery and Hovav Shacham. 2012. Pixel Perfect: Fingerprinting Canvas in HTML5. In *Proceedings of W2SP 2012*.

[51] 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)*.

[52] 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.

[53] 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.

[54] 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.

[55] Łukasz Olejnik, Gunes Acar, Claude Castelluccia, and Claudia Diaz. 2015. The leaking battery. In *Data Privacy Management, and Security Assurance*. Springer, 254–263.

[56] 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.

[57] Ashis Kumar Ratha, Shibani Sahu, and Priya Meher. 2018. HTML5 in Web Development: A New Approach. (2018).

[58] 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.

[59] 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.

[60] rovo89. [n. d.]. Xposed framework. https://repo.xposed.info. Accessed: 2018-06-14.

[61] 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*.

[62] 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.

[63] 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.

[64] 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.

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

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

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] 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.

[72] 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*. ACM, 113–124.

[73] 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.

[74] 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).