The Elephant in the Background: A Quantitative Approach to Empower Users Against Web Browser Fingerprinting

Anonymous

Abstract—**Tracking users is a ubiquitous practice in the web today. User activity is recorded on a large scale and analyzed by various actors to create personalized products, forecast future behavior, and prevent online fraud. While so far HTTP cookies have been the weapon of choice, new and more pervasive techniques such as browser fingerprinting are gaining traction. Hence, in this paper, we describe how users can be empowered against JavaScript fingerprinting by showing them when, how, and who is tracking them. To this end, we conduct a systematic analysis of various JavaScript fingerprinting tools. Based on this analysis, we design and develop FPMON: a light-weight and comprehensive fingerprinting monitor that measures and rates JavaScript fingerprinting activity on any given website in real-time. Using FPMON, we evaluate the Alexa 10k most popular websites to i) study the pervasiveness of JavaScript fingerprinting; ii) review the latest fingerprinting countermeasures; and iii) identify the major networks that foster the use of fingerprinting. Our evaluations reveal that i) fingerprinters are privacy-invasive and subvert current regulations; ii) they are present on many websites with sensitive contents (health insurance, finances, news, NGOs, etc.); and iii) current countermeasures can not sufficiently protect users. Hence, we publish FPMON as a free browser extension to empower web users against this growing threat.**

I. Introduction

Fingerprinting web users is a pervasive technique that lies at the intersection of good and evil. On the one hand, large amounts of device data are extracted by services to authenticate web users more securely and tackle the rampant amount of online fraud [38, 32], e.g., due to leaked credentials and automated attacks [19]. Likewise, large amounts of user data are used to optimize and improve web applications or study their usability [18]. On the other hand, the same device data is used to identify and track users across the web for targeted marketing and user profiling [17, 20].

To enable fingerprinting, various techniques have been created over the years to extract more and more device-specific data from the user's browser. At the same time, researchers have studied the various aspects of browser fingerprinting, especially after the first large-scale study by Eckersley et al. and the Electronic Frontier Foundation [9, 10]. Multiple studies have uncovered the prevalence of fingerprinting [1, 2, 9, 23, 41] by narrowing down on specific fingerprinting techniques, e.g., Acar et al. [1], discovered in 2013 that 5.5% of the 100k most popular websites use canvas fingerprinting and 1.5% of the 10k most popular websites use font-based fingerprinting [2].

Furthermore, other researchers have concentrated their efforts on creating new fingerprinting techniques, e.g. by using canvas [24] and audio objects [30] or via CSS [37]. Others have significantly improved known techniques with machine learning [42] or by extracting fingerprinting features more passively via extension activity tracking [36].

The sheer diversity of fingerprinting techniques available to us begs the question of which techniques are really being used and which ones are not? So far, large scale studies have only uncovered specific instances of fingerprinting and these are over 5-10 years old [1, 2]. In particular, we currently lack a comprehensive view of fingerprinting activity on the web today. This is an important point because when it comes to protecting users against such malicious techniques, countermeasures need to be designed according to this so that they can effectively block fingerprinting techniques.

To thwart such privacy-invasive behavior different stakeholders have adopted different solutions to the problem. Mozilla [26] and the most popular privacy extensions namely, PrivacyBadger and DuckDuckGo, use blacklists. Apple's Safari uses a simplified JavaScript API to reach some form of "herd immunity" and reduce the attack surface [4]. Blacklists have the major drawback that they are not effective against websites absent in the blacklist. Furthermore, the only means to establishing the effectiveness of such countermeasures is to obtain a report from privacy tools such as amiunique.org [35], panopticlick.eff.org [10], or browserleaks.com [5]. But, those tools do not cover all possible fingerprinting techniques and can not detect fingerprinting activity occurring on real websites visited by the user while browsing the web.

The situation is further exacerbated by the increasing functionality being introduced into browsers [34]. In fact, modern browsers give websites access to so many low-level device interfaces (GPU, Audio, USB, etc.) that it has opened the gates for high-precision side-channel fingerprinters [43, 31, 24]. This development is a particularly troubling matter as the lines between using these features for benign operations and tracking are very blurry. It may even prove to be impossible to make the distinction between the two and hence allow tracking without the user's consent even in the face of regulatory policy such as the General Data Protection Regulation (GDPR) or California Privacy Protection Act (CPPA) [15, 13, 6].

Motivated by i) the lack of a method to obtain a comprehensive and accurate estimate of all possible fingerprinting activity occurring on a real website and; ii) the inability of privacy tools to effectively thwart JavaScript fingerprinting as well as the lack of visibility into the presence of fingerprinting has led us to the creation of FPMON. FPMON empowers users to identify who, when and how JavaScript fingerprinting is executed on their devices. Rather than distinguishing fingerprinting based on single features we take a holistic approach to identifying fingerprinting as our view is that fingerprinting is more effective when features are combined. By classifying and rating an extensive JavaScript feature set, and quantifying the number of features accessed by a website, FPMON accounts for not only *any type of JavaScript fingerprinting activity but also any combination of fingerprinting features*. Our key idea to achieve this is to leverage popular fingerprinting tools to construct and classify a real-world feature set which can then be used by a browser extension to intercepts, analyzes and rate the JavaScript fingerprinting activity in real time. We believe FPMON has the potential to enhance the privacy of web users by making fingerprinting activity visible on every website and hence give people the power to uncover, understand, and discuss this emerging technology.

In this paper, we make the following **contributions:**

- We construct an extensive Javascript feature set based on real fingerprinting tools (closed- and open-source) that enables us to more techniques than any previous solution to the best of our knowledge.
- Based on a novel interception mechanism, we introduce FPMON to quantitatively measure and rate the presence of fingerprinting activity in real-time.
- Using FPMON we measure the widespread presence of obfuscated and concealed fingerprinting scripts for the Alexa 10k most popular websites as of March 2020. We conclude that roughly 19% of these websites collect user data via fingerprinting techniques without user consent.
- We demonstrate that most of today's popular countermeasures (Firefox, Privacy Badger, DuckDuckGo Privacy Ext.) are ineffective and explain why they fail to protect users against fingerprinting.
- We introduce a novel fingerprinting signature generation and matching scheme which when combined with FPMON enables us to identify the most prevalent networks that deploy fingerprinting scripts on the Alexa 10k most popular websites.
- Ultimately, we publish a browser extension based on FPMON. It can be accessed and installed in your browser at the below URL.

https://fpmon.github.io/fingerprinting-monitor/

II. Background

What is browser fingerprinting? Browser fingerprinting is the process of collecting a well-defined set of device features via the browser and generating a unique identifier, known as the *fingerprint*, of the user's device. The device features that are used to generate a fingerprint can be categorized as follows. *Technical features* relate to the software and hardware of the user device, e.g., screen size, CPU vendor, or memory size. *Sociocultural features* convey social, economic, geographic, and cultural information, e.g., languages, high-end or low-end device, timezone, etc. These features can either be *long-lived* and remain stable over time, e.g., browser vendor or content language or *short-lived* and change more frequently, e.g., browser version. The best features for fingerprinting offer a precise representation of the user's device and persist over time whereas features that change frequently are ill-suited. Regardless of what type of features is used, the uniform interface of JavaScript enables access to this data.

Feature	Example
User agent	Mozilla/5.0 (Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.130 Safari/537.36
Timezone	-60 (UTC+1)
Language	de, en-US, en, es
Fonts	Al Bayan, Al Nile, Al Tarikh, American Typewriter, Andale Mono & 182 more fonts
Plugins	Chrome PDF Plugin; Chrome PDF Viewer, Native Client, Flash, ...
Screen	1920 x 1080 x 24
Permissions	accelerometer:granted; camera:prompt; geolocation:prompt; background-sync:granted; magnetometer:granted; microphone:prompt;
Video format	video/mp4 flac; video/ogg theora; video/ogg opus; video/webm vp9, opus
Battery	charging; chargingTime: 0; level: 1
Connection	downlink : 10; downlinkMax : undefined; effectiveType : 4g; rtt : 100; type : undefined

TABLE I: A subset of device-specific features that can be collected with JavaScript and used to create a device fingerprint.

Fingerprinting with JavaScript: JavaScript (JS) is a just-in-time compiled programming language that is used to enable dynamic and interactive webpages in the World Wide Web. Therefore every browser has a dedicated JavaScript engine that executes the embedded scripts of a webpage on the user's machine. These scripts can interact with their environment, e.g., to adjust items to the screen size (`window.screen`) or show text in a specific language (`navigator.language`). By collecting large amounts of this device-specific data, a digital fingerprint of the device that runs the fingerprinting script can be created. A unique identifier for a device can be computed by hashing the concatenated data that is collected. Examples of what data can be used for this process are shown in Table I.

Advanced fingerprinting techniques: In addition to the naive approach of merely collecting device features, more advanced techniques that offer precise device identifiers also exist. In general, these techniques leverage the variations in hardware and software processing of the same instructions to generate a device fingerprint. For example, Mowery and Shacham [24] proposed a new fingerprinter that uses the JavaScript WebGL API. Using a 3D object in the browser and applying a set of textures and different ambient lights to it, the resulting picture slightly differs on every device thereby generating a fingerprint. Cao et al. [7] have shown that this technique alone can identify 99% of 1,903 tested devices. Another technique, called Canvas fingerprinting, is discussed by Acar et al. [1] and Englehardt et al. [12]. Hereby a specific picture is rendered with a fixed set of instructions using the HTML5 Canvas API. Depending on the operating system and the browser used, the created picture contains small variations. A unique identifier can be created by using the `toDataURL()` function to get a Base64 encoded representation of the rendered picture. Finally, a fingerprint can also be generated using the Web Audio API as described by Engelhardt et al. [12]. To use this, a sound signal of zero volume or the response of a dynamic compression applied on a sine wave is measured. A device-specific identifier can be derived by examining the resulting signal and compute a hash sum of this data.

III. The Threat of Browser Fingerprinting

To paint an accurate picture of the JavaScript fingerprinting ecosystem we first studied multiple public and private fingerprinting tools and several websites that use them. Among others, we examined the following fingerprinting tools: fingerprintjs.com, iovation.com, seon.io, datadome.co and sift.com. Based on this study, we have generalized the common business model and identified the main entities and their relationships, as illustrated in Figure 1. The fingerprinting ecosystem is composed of web users, content providers, fingerprinting tool suppliers, but also browser vendors and web developers. The main advocates that push the distribution of fingerprinting technologies are the tool suppliers and website owners that fuel the demand and want to better understand their users or secure their services. As shown in Figure 1, web users play only a passive role: they have no access to their profiles nor do they have the power to control how the data is used. The interactions between these parties can be described as follows:

1) The website owner embeds the fingerprinting script into the content of the website by embedding external scripts or adding inline code snippets.
2) When a user visits the webpage, the browser executes every script included in the loaded page source. As a result, the fingerprinting script executes and collects the device features.
3) Either all the collected data is sent to the fingerprinting or a unique identifier, e.g. a hash value, is created and sent to the fingerprinting service provider.
4) The service provider matches the received identifier against a database of known profiles. Either a profile matches or a new profile is created in the database.
5) In the end, the website owner can either access the results of the analysis or immediately receives insights, e.g., the user can be trusted or not. The service provider is paid by volume, license, or monetize the service in other ways.

In reality, many things can differ from our generalized model. For example, a website owner can unwittingly add a fingerprinter to its website by adding a 3rd-party plugin that includes a fingerprinting script. In other cases, the data is sent directly to the content provider, and not to a 3rd-party service. However, we never know if this data is shared with 3rd-parties via backend communication later on or not [22].

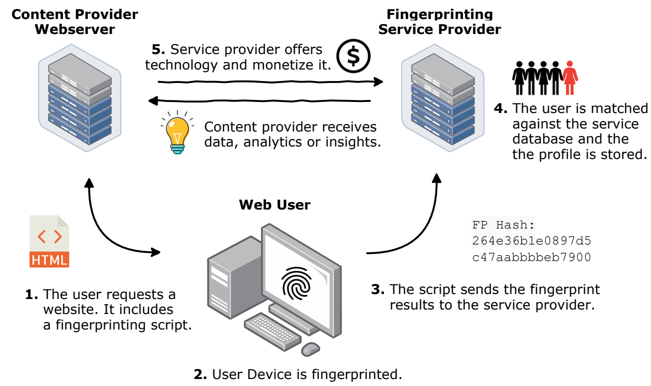

Fig. 1: The fingerprinting ecosystem: Users access content that embeds fingerprinting scripts from 3rd-parties; those collect, analyze and monetize the user profiles.

Threat Model: We assume that some webpages contain fingerprinting scripts that extract user data to construct a model of the user. Depending on the perspective, this model can be used for benign or malicious applications, e.g. user tracking, targeted advertising, product improvements, or better security. We assume, not every website includes fingerprinters intentionally or follows the purpose of user identification directly, even if the data collected will allow this. In other words, we assume that websites have been deployed to respect user privacy laws such as GDPR or CCPA. We assume users do not want to be tracked and identified without their consent. Specific user groups will deliberately use protections, e.g. privacy extensions, to circumvent tracking and user profiling. According to previous research, only 2% of users disable JavaScript completely, which will break most of today's web functionality and hence is not a practical solution [44]. With respect to detecting fingerprinting, it is impossible for the client to say that the data extracted via JavaScript is definitely used for fingerprinting. Hence, we assume that a website that uses more features than necessary for its application is conducting some form of fingerprinting. We elaborate on this in Section IV.

Fingerprinting Protection: To protect users against fingerprinting major browser vendors such as Mozilla and Apple have introduced countermeasures. Mozilla's Firefox offers a feature that is intended to block fingerprinting and crypto-mining scripts by blacklisting domains that serve fingerprinting scripts [26]. Apple's approach to combat fingerprinting in Safari is different and is promoted as a type of "herd immunity" [4]. By presenting a simplified version of the system configuration, such as installed fonts and plugins, to trackers it makes more devices look identical. This reduces the capability of fingerprinters to identify a single device without breaking web functionality [4]. For Google's Chrome Browser, the most popular one in recent years, Google announced to block fingerprinting in the future [29]. So far, it's unclear how their solution will work. As long as no protections are available, privacy extensions such as the EFF Privacy Badger and the DuckDuckGo Privacy Extensions can be installed for every major browser [14, 8]. These extensions follow the same approach as Mozilla, which is to blacklist domains that are known to be privacy-invasive. The ultimate solution to thwart JavaScript fingerprinting is to disable JavaScript completely. However, this is not a practical solution as it breaks most of today's web functionality and less than 2% of users do this [44]. Academics have identified this problem and a couple of solutions have been recently proposed. In 2019, Wu et al. proposed a uniform shader language execution to prevent WebGL fingerprinting [43] and Trickel et al. introduced to mitigate the fingerprinting of browser extensions [39].

IV. Classifying Fingerprinting Features

The first step on our mission to empower users against browser fingerprinting requires us to understand and classify the JS functions that are typically used to fingerprint a device. To this end, we have systematically analyzed the commercial and public fingerprinting tools mentioned in Section III as well as the following privacy tools amiunique.org, panopticlick.eff.org and browserleaks.com. By reverse-engineering the proprietary and obfuscated tools using Chrome DevTools [16] and the Burp Web Security Suite [28] we obtained a collection of 115 JavaScript functions and properties that were used by the fingerprinting tools. Indeed not every JS function is individually responsible for fingerprinting, but when combined in a specific order, these functions are indicative of fingerprinting activity.

Next, we classified the 115 functions and properties into features based on the capability of the functions to analyze a certain device feature. This classification yielded 40 features where each feature represents an individual vector to fingerprint a user, e.g., Screen, Window, Language, etc. as shown in Table II. The complete set of features are shown in the X-axis in Figure 5. We note that features can contain multiple ways to perform the same operation because of browser differences or syntactic shortcuts, e.g., DoNotTrack and msDoNotTrack as shown in Table II. Since our analysis is based on the tools we analyzed, our specified feature set may miss single functions that are used by fingerprinting tools that we did not study. Nonetheless, the feature set is designed to represent real vectors and not hypothetical corner cases.

To account for the different capabilities of the feature, we apply simple weighting mechanism by labeling each feature with a *severity rating*: sensitive or aggressive. The rating is calculated based on the similarity ratio from amiunique.org [35] and the entropy research from Panopticlick [11, 10]. We determined the severity of each feature in three steps: i) we tagged all features that contain functions with a similarity ratio $\leq$ 30% ii) we tagged all features that contain functions with a an entropy value $\geq$ 5 bits iii) each feature that have been tagged twice is rated aggressive, all others are rated sensitive. If we could not obtain a similarity ratio or entropy value for a feature, we estimated the rating based on the type or quantity of data accessible via the feature under question. We decided to not weight each feature individually to reduce the risk of over- and underweighting of the features with unequal cardinality.

Sensitive features have high similarity ratios (>30%) and low entropy (< 5 bit) making them inaccurate when used individually. They are necessary to enhance the user experience, e.g., to show the correct language or current time. To identify a user with high accuracy, a fingerprinter needs to use many sensitive features simultaneously.

Feature	JavaScript functions	Rating
Screen	colorDepth, width, height, availWidth, deviceXDPI, ...	sensitive
Window	devicePixelRatio, innerWidth, colorDepth, outerWidth, ...	sensitive
Flags	doNotTrack, msDoNotTrack	sensitive
Audio	createAnalyser, createOscillator, createGain, createScriptProc., ...	aggressive
Language	languages, userLanguage	sensitive
Storage	sessionStorage, localStorage, indexedDB, openDatabase, ...	sensitive
Battery	getBattery, charging, ...	aggressive
WebGL	getShaderPrecisionFormat, shaderSource, createBuffer, ...	aggressive

TABLE II: Examples of JavaScript functions with their associated feature name and rating. In total, we classified 40 different features for 115 individual JS functions.

Aggressive features have low similarity ratios ($\leq$ 30%) and high entropy ($\geq$ 5 bit) making them precise. They can generate a high bit of entropy and make use of browser functionality that is questionable for most application, e.g., battery level or audio oscillators. Some aggressive features can potentially identify users with high precision and do not necessarily need many additional features to do so, e.g., Canvas and Audio.

Clearly, not every classified JS function is directly related to fingerprinting. More importantly, it is fundamentally impossible for a user who visits a webpage to know whether she is being fingerprinted or not unless it is explicitly stated. Hence, we argue that the combined use of the JS function set is a strong indicator of fingerprinting activity, especially when several aggressive features are used. When a website uses many of the sensitive and aggressive features in a particular composition and in a short time frame, it becomes likely that a fingerprinter is active on the website.

V. FPMON - A Fingerprinting Monitor

Having studied existing tools and classified the JavaScript features, our next step is to design and develop a tool that can record and analyze all the classified features. The high-level workflow on how this can be done is described in Figure 2.

Idea and benefits: The core idea of FPMON is to dynamically add an interception mechanism in front of the classified JS functions before the real webpage context is executed. By modifying the JavaScript runtime environment with code injections, we can intercept and record the functions without altering the default runtime behavior. The major benefit of this approach is browser independence, i.e., the fingerprinting monitor can be easily imported into any up-to-date browser.

Challenges: To realize our idea, we have to first overcome three main challenges. First, the JavaScript runtime environment needs to be modified on-the-fly and without hampering functionality, e.g., the page should still be rendered correctly. Second, the evaluation of the recorded data needs to be quick and light-weight, so that the results can be displayed as soon as the page is loaded. Third, the results need to be communicated in an intuitive manner, i.e., it should not require technical expertise, background, or a particular language. In

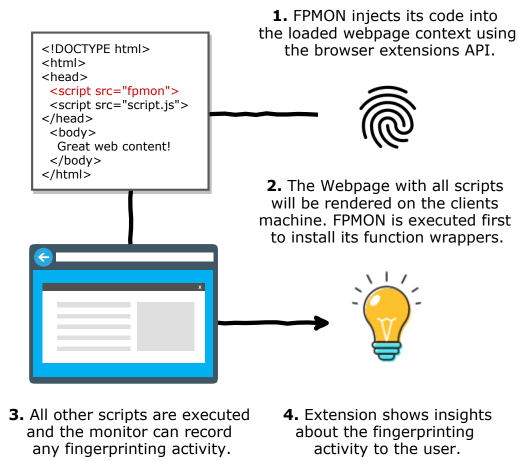

Fig. 2: The workflow of FPMON to measure and analyze the fingerprinting activity of a webpage.

the following, we elaborate on how we address the above challenges, our system architecture and inner workings of FPMON as well as the applications that have been built on top of this.

A. Intercepting Feature Access

The interception mechanism of FPMON is similar to a man-in-the-middle proxy in that it i) observes and records every function access attempt; and ii) it is transparent to the function caller which preserves runtime behavior. Since there is no such interface, we developed a mechanism to intercept all the classified JS functions discussed in Section IV. This proxy monitors every access attempt and intercepts each function call and its associated arguments and return values, completely during run-time.

To implement this proxy in JavaScript, we made use of the `defineGetter()` operation of the object prototype. Using that operation, we can override an object's property to execute our custom code in addition to the original functionality. The custom code performs two main tasks: i) it records the object that is accessed; and ii) it records the return values that are passed to the function caller at the end. By implementing this for various JavaScript objects, e.g., Window, Audio, WebGL, Canvas, and many of their sub-properties we can track and examine each individual function call with its arguments and return values. In addition, we can record the script host and filename by tracing the call stack via `Error.stack` which can aid in pinpointing the script source of suspicious behavior.

To ensure that we capture all JavaScript executions beginning with the first script embedded into the webpage, we introduce the following two steps. First, we load the monitor script code into the page context with the help of the browser extension API. Second, we design the monitor as a self-executing script to ensures that the intercepting functions are in place before anything else is executed. In particular, we use the `content_script` option to execute our script at `document_start`. This content script injects the monitor code that modifies the webpage's load process as follows:

Algorithm 1 Rating the Fingerprinting Activity

Require: features_{all} and features_{aggro}, amount of all features, and all aggressive features enabled on a webpage.
  rating = low
  if $\text{features}_{all} > 27\%$ **or** $\text{features}_{aggro} > 16\%$ **then**
    rating = medium
    if $\text{features}_{all} > 42\%$ **or** $\text{features}_{aggro} > 33\%$ **then**
      rating = high

1) The webpage is loaded from the server and the FPMON script will be injected via the browser extension API.
2) When the browser attempts to render the page, the injected code is executed first and overwrites the original JavaScript functions with our wrapper functions.
3) Afterwards, every script from the webpage that tries to access a monitored function will unintentionally call the wrapper function, which logs the access, executes the intended operation, and pass through the return values.
4) Finally, the recorded results are passed back to the browser extension for the subsequent analysis and rating.

B. Rating of Fingerprinting Activity

After the webpage is loaded we analyze the collected data and rate the page the user has just visited. In particular, FPMON rates the page based on a quantitative measure which is described as follows. If a page script uses one of the functions that are assigned to a feature, we register this feature as enabled. For every enabled feature, the page score increases by one. Recall Section IV, we identified 115 functions and classified them into 40 distinct features. Hence, for the quantitative measurement we perform three steps: i) count the number of functions accessed; ii) enumerate how many of the 40 features are enabled and; iii) check how many of these features are labeled aggressive. To translate the intermediate results into a final score, we need to apply appropriate thresholds as shown in Algorithm 1. The threshold values are based on our evaluation of the Alexa 10k popular websites (Section VI-C). In particular, using data from the evaluation, we calculated the median fingerprinting activity and the mean absolute deviation (MAD) for all features, including the aggressive ones. Based on how much activity we find to be normal on a relative scale, we rate the webpage behavior to be low ($\leq$ median), medium ($\leq$ median+MAD), or high ($>$ median+MAD). In Section VI-C, we describe how we obtain concrete threshold values based on empirical data from the 10k most popular websites.

There are two main weakness with such a methodology. First, FPMON does not monitor all JS functions related to fingerprinting (recall Section IV), hence, some features may go undetected. Second, if special conditions need to be satisfied to trigger fingerprinting, FPMON will falsely rate a page even though fingerprinting conditionally occurs on the page. For example, i) fingerprinting is deactivated for users with valid cookies (see Section VI-A); and ii) enable fingerprinting only on specific pages, e.g., login pages or business client portals. On the positive side, a single feature used in a benign

Fig. 3: The icons to indicate the fingerprinting score in the browser extension. The color guides the user to understand if a page scores low, medium or high. The badge text tells the user how many of the tracked features are accessed.

way will falsely influence the overall rating by only a small fraction (1/40). Likewise, a missing component in the monitor can distort the result by only a small fraction (< 1/40). We believe these properties make our approach very resistant against false positives, compared to previous solutions that only discriminate based on a single feature [2, 1]. As we will see from our evaluations on real websites in the next sections, our measurements based on this scheme will provide a conservative estimate.

C. Empower Users Against Fingerprinting

Leveraging the monitoring logic presented above, we created a comprehensive browser extension that can be used to evaluate one's favorite websites. The extension is built for Chrome, but can be imported easily into any browser supporting a similar extension API such as Safari and Firefox. For the extension, we designed two visualizations to present the results to the user. First, we created a simplified way to signal the ongoing fingerprint activity. Depicted in Figure 3, we designed a browser extension icon that shows a human fingerprint. The color of the icon changes according to the level of fingerprint activity detected, i.e., based on the thresholds described in Algorithm 1. In addition, the total number of detected features is also shown using the icon badge text. Second is a technical view of the overall analysis that is only visible to the user if the extension icon is clicked on. In this view, we summarize the exact number and the name of all JavaScript features that have been enabled on the loaded webpage. In addition, we list the top 3 script files that enabled most fingerprinting features to aid the user to find out which scripts are analyzing them. To create the user extension, FPMON was extended with a lot of functionality to encode, transmit, and update the data that is shown in the user interface. The extension can be installed via the URl specified at the end of Section I.

VI. Evaluation

We now shift our focus to evaluating the effectiveness of FPMON in monitoring fingerprinting on real websites. Hence, we designed a set of experiments to answer the following questions: i) how widespread is the use of JS fingerprinting and in what context can we find it; ii) how effective are existing countermeasures; and iii) what are the most prevalent networks that foster the use of fingerprinting? To answer these questions, we conducted two types of studies. In the first, we curated a set of 20 websites that we analyze in great detail and in the second, we conduct a large scale evaluation on the 10k most popular websites listed by alexa.com [3]. In the first study, the small data set allows us to carefully evaluate FPMON to obtain a deep and detailed understanding of the context of JS fingerprinting and how it is applied. In our second study, we carry out a large scale evaluation of the 10k most popular websites to statistically describe the fingerprinting landscape and investigate the widespread use of this technology. Furthermore, the large data set also enables us to uncover any networks that may exist behind the fingerprinting scripts. In the following, we elaborate on each of those studies.

A. Real-World Abuse of JavaScript Fingerprinting

In this study, we evaluated a self-curated list of 20 popular websites that cover a large range of representative topics: financial services, online search, news, file-sharing, governments, NGOs, healthcare, pornography, etc. The objective of this preliminary set of websites is to show how FPMON can identify the presence of fingerprinting scripts in detail and to evaluate the effectiveness of fingerprinting countermeasures introduced in Section III.

Methodology and setup: For each website, we recorded the number of functions used by the fingerprinting script in a database. Based on the collected data, we counted the number of enabled features, the number of aggressive features (recall Section IV) and afterwards calculated the final score for each webpage. The score shows how many fingerprinting features are enabled for each website and is measured relative to the total number of monitored features. For repeatability, we visited every website multiple times. Almost every page consistently scored the same. For a few cases, the results have varied slightly (within ± 5% score), which we did not investigate further. For the sake of easy interpretation, we only show the scores from individual runs. The data was recorded on 13. April 2020. We note that results can change over time due to changes on websites. For the first part of the evaluation, we used Chrome version 81 with default settings that was extended with the FPMON browser extension. The experiments are executed on a 2017 MacBook Pro.

Results: Table III shows the data we collected from our first study. Starting from left to right, we listed the fine-grained results for each of the websites tested. The functions detected column shows the number of functions that are used by the webpage. Next, we show how this relates to the enabled features and how many of those are considered aggressive. Next, is the final score depicted as, ○, ◐, and ● which correspond to the simplified score of FPMON annotated with the relative score calculated. The data is sorted in descending score order for ease of reading.

Baseline comparison: Our baseline is represented by panopticlick.eff.org. The Panopticlick website is a privacy test and measures if one can be uniquely identified based on the data extracted from the browser. As shown in table III, panopticlick covers around half of the functions (64/115) that are monitored by FPMON. These 64 functions relate to a total score of 53%, because 21 features are enabled of which 10 are labeled aggressive. Similar services, namely fingerprintjs.com and amiunique.org achieve very similar scores of 48% and 58% respectively. Although these websites already cover many fingerprinting techniques, we note that they fall short on several functions that FPMON covers (almost half of the features) and which are actually used by higher-scoring websites.

Domain	Page topic	Functions detected	Features detected	Aggressive Features	Chrome Score	Chrome + AdBlock	Privacy Badger	DuckDuckGo Privacy Ext	Firefox Standard	Firefox Strict	Apple Safari	Protection working?
metacafe.com	video sharing	95 / 115	38 / 40	17 / 18	● 95%	● 95%	● 95%	● 95%	● 95%	○ 18%	● 50%	✗
nypost.com	news media	66 / 115	35 / 40	14 / 18	● 88%	● 43%	● 45%	● 43%	● 48%	◐ 40%	○ 18%	✗
addthis.com	tracking tool	62 / 115	34 / 40	13 / 18	● 85%	○ 20%	○ 18%	○ 5%	○ 23%	○ 23%	○ 18%	✓
nasdaq.com	stock data	53 / 115	29 / 40	12 / 18	● 73%	● 70%	● 70%	● 73%	● 45%	◐ 35%	● 45%	✗
easyjet.com	flight booking	62 / 115	27 / 40	11 / 18	● 68%	● 50%	● 50%	● 48%	● 50%	● 48%	● 45%	✗
bankofamerica.com	financial services	58 / 115	25 / 40	8 / 18	● 63%	● 63%	● 63%	● 63%	● 55%	● 55%	◐ 35%	✗
nytimes.com	news media	49 / 115	24 / 40	12 / 18	● 60%	● 60%	● 60%	○ 18%	● 60%	● 60%	◐ 38%	✗
coinbase.com	crypto exchange	68 / 115	23 / 40	11 / 18	● 58%	● 58%	● 58%	● 58%	● 58%	● 58%	○ 25%	✗
savethechildren.org	non-profit orga.	72 / 115	23 / 40	11 / 18	● 58%	● 58%	◐ 35%	○ 20%	● 58%	● 58%	● 45%	✗
alibaba.com	e-commerce	64 / 115	21 / 40	9 / 18	● 53%	● 53%	● 53%	● 53%	● 53%	● 53%	◐ 28%	✗
panopticlick.eff.org	privacy test	62 / 115	21 / 40	10 / 18	● 53%	● 53%	● 53%	● 53%	● 53%	● 53%	◐ 28%	✗
healthcare.gov	healthcare	43 / 115	20 / 40	10 / 18	● 50%	● 50%	● 43%	◐ 40%	● 48%	● 45%	◐ 35%	✗
vyprvpn.com	privacy tool	64 / 115	19 / 40	6 / 18	● 48%	● 48%	○ 25%	○ 23%	○ 25%	○ 25%	◐ 30%	✓
theguardian.com	news media	41 / 115	15 / 40	4 / 18	◐ 38%	◐ 30%	◐ 28%	○ 23%	◐ 33%	○ 15%	○ 15%	✓
google.com	search engine	22 / 115	13 / 40	7 / 18	◐ 33%	◐ 33%	◐ 33%	◐ 33%	○ 25%	○ 25%	○ 15%	✓
pornhub.com	pornography	19 / 115	9 / 40	2 / 18	○ 23%	○ 18%	○ 18%	○ 18%	○ 23%	○ 18%	○ 15%	-
wikipedia.org	encyclopedia	12 / 115	7 / 40	0 / 18	○ 18%	○ 18%	○ 18%	○ 18%	○ 18%	○ 18%	○ 13%	-
nsa.gov	security agency	11 / 115	6 / 40	2 / 18	○ 15%	○ 15%	○ 10%	○ 10%	○ 15%	○ 10%	○ 13%	-
europarl.europa.eu	goverment	15 / 115	5 / 40	0 / 18	○ 13%	○ 13%	○ 13%	○ 13%	○ 8%	○ 8%	○ 5%	-
torproject.org	anti-censorship	4 / 115	1 / 40	0 / 18	○ 3%	○ 3%	○ 3%	○ 3%	○ 3%	○ 3%	○ 3%	-
wikileaks.org	whistleblowing	0 / 115	0 / 40	0 / 18	○ 0%	○ 0%	○ 0%	○ 0%	○ 0%	○ 0%	○ 0%	-

TABLE III: Calculated fingerprinting scores for popular websites from different topics. On the left half, we list the intermediate and final results of FPMON. Higher numbers indicate that high number of fingerprinting techniques were detected. ○, ◐, ● indicate if the webpage is rated low, medium, or high. In the right half of the table, we see the results of the tested countermeasures. The last column summarizes if at least one solution for each browser can reach a medium or low rating (✓) or not (✗).

The scoring spectrum: The highest score measured is 95% on metacafe.com. The website uses 95/115 monitored functions, which relates to 38/40 features including 17 aggressive ones. Next, we found many websites with privacy sensitive content that also make use of a high number of fingerprinting features. For example, financial service websites such as bankofamerica.com (63%), nasdaq.com (73%) and bloomberg.com (68%) using more than half of the aggressive features we monitor. News and media websites such as nypost.com, nytimes.com and wsj.com also yield very high scores, 88%, 60% and 58% respectively. Equally alarming are the results for health insurance services like healthcare.gov (48%) and medicare.gov (53%). Furthermore, a lot of device data is also collected by more privacy promising organizations like coinbase.com (58%) or vyprvpn.com (48%). In contrast to this, websites such as wikipedia.org and europarl.europa.eu receive very low scores and seem to respect their users' privacy: they do not use aggressive features and limit the number of device information extracted. Websites that scored the least are torproject.org and wikileaks.org, they hardly use any of the monitored features. Ultimately, we note that many of the websites reach a similar or higher score than the baseline (approx. 50%) [35, 10]. If a user can be identified in the baseline case, it appears likely that user can also be identified by another entity collecting a similar amount of device data. In contrast, other websites offer similar contents without the need for this amount of data.

User consent and dealing with cookies: A key observation from our study is that *most websites extract device data even before the user accepts a cookie banner or agrees to any kind of privacy policy*. In our opinion, this behavior is very problematic when considering regulations such as the General Data Protection Regulation (GDPR) introduced by the European Union in early 2018 [13]. As argued by the EFF, browser fingerprinting clearly falls under the broad definition of personal data [15], hence, the observed behavior appears to subvert GDPR regulations. The only website we found to respect the user's consent is addthis.com. Although it is not so easy to adjust the privacy settings, fingerprinting only occurs if the user allows the page to do so (85% vs. 8% score). Another noteworthy observation was made on nasdaq.com. The fingerprinting on this website is much stronger if the user is not recognized by a cookie (73% vs. 38% score). In fact, only two of our test cases behave differently when a user provides a valid cookie.

Key takeaways: The key takeaways from this evaluation are i) JavaScript fingerprinting is used and applied in many sensitive contexts where privacy is important; and ii) many websites seem to disrespect the user's consent and hence might subvert current privacy regulations.

B. *Effectiveness Of Privacy Countermeasures*

Having observed the dominance of JavaScript fingerprinting on various websites, we evaluated whether privacy tools and fingerprinting countermeasures are effective against this threat. The majority of tools blacklist (known) tracking providers to block content and functionality from their domains, including fingerprinting scripts.

Methodology and setup: We followed the same methodology described previously (Section VI-A), hence, here we limit our description to the countermeasures. Using Chrome we evaluated the EFF Privacy Badger [14] and the DuckDuckGo Privacy Extension [8]. Both extensions claim to protect users against fingerprinting. In addition, we included the AdBlock extension in our evaluation to i) show how a default ad blocker performs in comparison and; ii) see how much fingerprinting is introduced by ad networks. For Mozilla Firefox, we evaluated Mozilla's Enhanced Tracking Protection with *standard* and *strict* settings. According to Mozilla, the *strict* mode is meant to offer more privacy and block fingerprinting and other tracking techniques less carefully which might break some web functionality [25]. For Apple Safari, we used an out-of-the-box Safari browser that includes Apple's fingerprinting protection by default [4]. For a fair comparison, we firstly had to evaluate how well Apple's unification approach works [4]. Hence, we compared the data collected by four fingerprinting tools ([35, 40, 5, 10]) across three different Apple devices to find all the unified functions. We found 11 functions that are not supported by Safari, e.g., CPU, memory, or battery information and 20 that do not differ across the devices. However, numerous functions still return different data across devices such as user agent, screen, timezone, language, and even so the list of fonts and plugins. We also found that some advanced fingerprinting techniques have either been thwarted or never worked for Safari. For example, there is different audio and WebGL context data (e.g. different GPU vendors), but the audio signal and WebGL image hashes are the same across all devices in our test. For a fair comparison, we did not track the functions that are unified by Apple and hence look identical across all tested devices. In total, we did not track 31 functions out of 15 features that are monitored by FPMON. The total number of functions and features used to calculate the results with FPMON is the same for all experiments. The data was recorded in the third week of April 2020 using the following browser versions: Chrome 81.0, Firefox 75.0, Safari 12.1.

Results: On the right side of Table III we present the results from our evaluation. As with the page score, the higher the value, the more fingerprinting features are accessed and the more data is extracted. Starting from the Chrome score, what follows are the results for the Chrome privacy extensions. Following that, we have Firefox in the standard and strict mode. Next, we have Apple Safari and finally, a column to summarize if at least one solution for each browser can reach a medium or low rating. In general, an effective protection should considerably reduce the number of features in absolute terms and in comparison to an unprotected browser. The Chrome score can serve as a reference since each solution has been evaluated based on the same number of features. In absolute terms, it is not clear how much and what type of data is required to identify a user. However, by reference to our baseline (panopticlick.eff.org) we assume that scores of around 50% and more will facilitate the identification of users with high probability. Depending on the underlying model and the feature types, fewer or more features can be required.

Privacy Extensions for Chrome: For the three privacy extensions installed in Chrome, we can see that the scores for many websites remain the same whereas the score for some reduced drastically. For example, the extensions reduced the scores for nypost.com, addthis.com, savethechildren.org and vyprvpn.com by more than half or even three-quarters. Although the score reduced by a considerable amount for some websites, the reduced score remains high and is still close to the baseline (e.g., nypost.com) which means that fingerprinting is still feasible. For more than 30% of the tested websites, the protections did not impair fingerprinting. A noteworthy result here is that the AdBlock extension produced results similar to the two other privacy extensions we evaluated by merely blocking 3rd party ad networks. In summary, both privacy extensions give users a false sense of security in the context of fingerprinting. Although the extensions may be useful against other ways of tracking, they are not sufficiently effective against fingerprinting. Finally, the results from evaluating AdBlock suggest that invasive fingerprinting code occasionally occurs via 3rd-party ad networks.

Mozilla Firefox: To begin with, Firefox has a slightly smaller attack surface compared to Chrome, since a few functions are not supported. In total numbers, Firefox standard perform similar to Chrome with Privacy Badger . When it comes to Firefox in the strict mode (highest privacy setting), we observe that only in 7 out of 15 relevant cases shown in Table III, the fingerprinting has been thwarted. For approximately half of these cases, the user is not protected at all and can potentially be identified with the amount of data extracted.

Apple Safari: We observe that the websites score considerably lower when loaded via Safari. The total page score, for all test cases, has been nearly halved (47,8% less) compared to an unprotected Chrome. In addition, for many webpages we tested, the scores were far below the 50% baseline of panopticlick.eff.org. However, in 4 out of 15 relevant cases shown in Table III, the number of fingerprinting features accessed might still be problematic. Despite the great efforts, many fingerprinting techniques still work, e.g., canvas fingerprinting with PNGs and also the general collection of all features. This explains why fingerprinting tests such as amiunique.org can still uniquely identify our full fingerprint among their two million collected profiles. In summary, the limited feature set available in Safari coupled with Apple's unification strategy reduces the possibilities to fingerprinting users significantly. While the protection is not bulletproof, it becomes provable more difficult to identify individual users.

Key takeaways: In summary, the popular privacy extensions and Firefox do not sufficiently protect users from being profiled via JavaScript fingerprinting. However, they still offer protections against other forms of tacking and cryptomining. Furthermore, Apple has implemented a simple yet effective approach: Reducing and unifying the JavaScript interface to considerably protect the user against the various fingerprinting techniques.

C. Large-Scale Website Analysis

The objective of this study is to evaluate the presence of JavaScript fingerprinting and to what degree it is present on popular websites. Hence, we crawled the 10k most popular

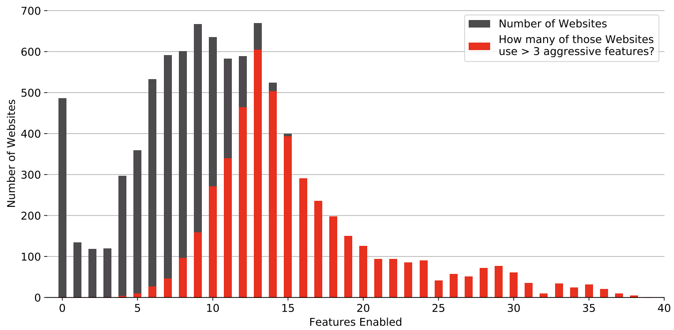

Fig. 4: Distribution of enabled features and how many websites use more than 3 aggressive features (red).

websites listed by alexa.com with FPMON to demonstrate how it can be efficiently used for large-scale fingerprinting analysis.

Methodology and setup: To automate the experiment we designed and built a crawler to scan the 10k most popular websites from alexa.com in early 2020. The crawler scans the list of websites within a dockerized chrome and is controlled via the selenium framework in python. Each browser instance is loaded with a modified version of our FPMON extension that can handle timeouts and sends the collected data to a local server that stores the data in a database. Each browser instance uses a clean and new profile for each website crawled. It was configured with a 45 seconds implicit timeout and a maximum timeout of 90 seconds, after which the chrome process terminated. We greatly improved the scanning process over time by adding signaling between browser and crawler to handle corner cases (timeouts, HTTP errors, etc.). A scan of the 10k websites using 20 parallel dockerized chrome instances takes about 5.5 hours on our dedicated server setup using an AMD EPYC 7272 12-Core CPU, 48 GB RAM, and a 1 Gbps uplink. The full results are available in our GitHub repository.

Results: The websites we scanned are listed as the 10k most popular websites by alexa.com in early 2020. We successfully collected data from 9,192 pages: we did not receive responses from 674 pages for various reasons, e.g., country restrictions. In addition, we removed 134 obvious duplicates that occurred due to language and protocol redirects. In Figure 4 we show the distribution of website scores and in Figure 5 we show the distribution of features used across the websites we crawled.

Baseline comparison: As in Section VI-A, we can use panopticlick.eff.org as a baseline reference. In total, we found 9.66% of the 10k websites use more than or a similar amount of fingerprinting features compared to this fingerprinting demo, which can uniquely identify users with high probability.

The scoring spectrum: The highest scores (see Fig. 4) are achieved by breitbart.com, foursquare.com and politifact.com: each of them used 36 features (95% score) by calling around 100 JavaScript features and using nearly all the aggressive features we classified. In our data set, no website was found that uses all the features, however, we observe that each feature is indeed used several times. On the lowest end of the distribution, around 5.3% of all websites do not use any kind of fingerprinting feature that we monitor. Moreover, we found that the majority of websites (56.8% of all websites tested) lie at the center of this distribution and use 11 ± 4 features. The 1st, 2nd and 3rd quartiles of the distribution are 7, 11, and 14 features respectively. The median amount of features is 11 with an absolute deviation of 5.2. Accordingly, we found i) 55.72% of the websites use 11 or fewer features; ii) 26.89% apply 11-16 features; and iii) 17.38% use 17 or more features.

Aggressive feature usage: From a privacy perspective, it is the use of aggressive features that is concerning (recall Section IV). As illustrated in Figure 4, the number of aggressive features is lower and less frequent compared to the non-aggressive features. We calculated the global median usage of aggressive features to be 3 with an absolute deviation of 2.49. Accordingly, we found i) 62.16% of the websites use a maximum of 3 aggressive features; ii) 23.22% use 4 or 5 aggressive features; and iii) 14.62% websites use 6 and more aggressive features. Looking in particular only at websites that are rated low, medium, or high by FPMON, we find the average amount of aggressive features used is 1.34, 3.5, and 11.47 resp. Hence, websites that generally collect lots of user data, also tend to use a notable amount of aggressive features.

Determining thresholds for FPMON: Based on the distribution of features we collected, we adjusted the thresholds that are used by FPMON to rate a webpage. As shown in algorithm 1, we distinguish between aggressive and non-aggressive features because their distributions are so different. We rate the recorded fingerprinting activity of a webpage as low if the number of features is $\leq$ the median behavior. A website is rated medium if it scores above the median but is still below the upper bound of the absolute deviation (Median+MAD). Every website scoring above this range is rated high. Accordingly, we found 52.90% of the websites score low, 28.09% medium, and 19.01% high. We discuss the tradeoffs of this approach in Section VII.

Distribution of JS features: Figure 5 shows the distribution of all the features that FPMON tracked for the 10k websites. We can draw two main observations from this data. First, it is clear that four features namely, User-Agent, Screen, Content Language, and Plugin List, are used by nearly all the websites regardless of their fingerprinting score. Second, websites with high fingerprinting activity (red) in contrast to websites with low activity (green) use more features: We find that around 20 features are used by websites with high fingerprinting activity, but they are almost never used by low scoring websites. Hence, we strongly question the use of those features on such websites as they do not seem to serve a benign purpose. For example, why do these websites need to know specific GPU, memory, connection, and battery information about the user's device?

Font fingerprinting: In 2013 researchers summarized that close to 1.5% of the top 10k websites track users using font fingerprinting [2]. Our data shows that 1,360 pages have a medium or high rating and use font fingerprinting features via JavaScript. This relates to 14.79% of the 10k most popular websites. Hence, we estimate a 10x growth in the use of font fingerprinting within the last 7 years.

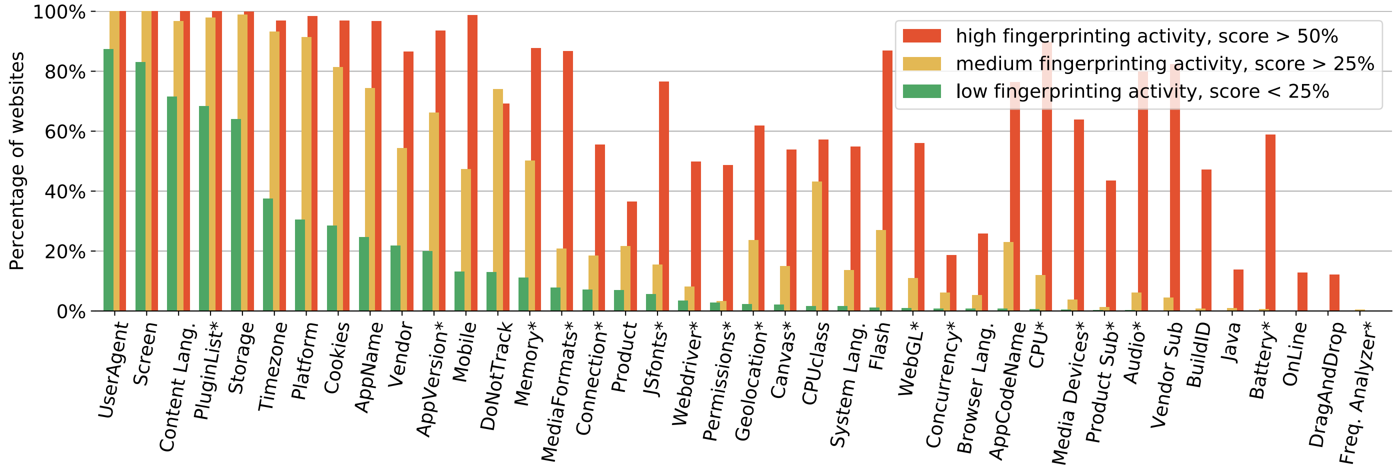

Fig. 5: Distribution of feature usage for websites with low, medium and high fingerprinting activity. (* = aggressive features)

Canvas fingerprinting: In 2014 Acar et al. [1], analyzed the most popular 100k websites and concluded that around 5.5% of websites apply canvas fingerprinting. According to our data, we found 1,641 websites with a medium or high rating that make use of Canvas fingerprinting features, which relates to 17.85% of the 10k pages. Hence, we approximate a 3x growth in about 6 years.

Top level domains: We filtered the data by top-level domains (TLDs) to calculate the average scores by country and types of organizations. For fairness, we removed underrepresented TLDs with less than 40 entries and tested for sample size issues. The top scoring TLD was `.ru` with an average score of 15.7%, followed by `.uk`, `.vn`, `.au`, `.de` and `.br` all of which scored approx. 13.2%. The most popular `.com` domain, is in the mid-range of this ranking with an average score of 11.8%. The `.gov` TLDs have an average score of 10.6% and the .org domains score 9.7%. Both can be found in the last third of the ranking. One of the lowest scoring TLDs in the list is the `.edu` domain with an average score of 8.1%, almost half as much as the top-scoring TLD. The ranking illustrates that JS fingerprinting is i) most common on Russian domains; and ii) more adopted by private entities compared to public entities such as governments, NGOs, and universities.

Key takeaways: Our conclusions from analyzing the 10k most popular websites with FPMON are the following. First, fingerprinting has grown tremendously in the past 5 years. Second, around 19% of websites make massive use of fingerprinting features and approx. 28% of websites collect an above-average amount of user device data. Third, around half of the features we identified are questionable and tend to be used against the interest of users.

D. Fingerprinting Networks

In our final study, we aim to answer the following questions: i) What are the most aggressive fingerprinting scripts? ii) Who are the main distributors of these scripts? And iii) How widespread are their networks? We do so by investigating the individual fingerprinting scripts from our data set and looking for common patterns in their behavior.

Methodology: Based on the data collected on the 10k most popular websites, we consolidated the data for each of the occurring scripts in the following way. Instead of a page score we calculated a script score. The score still indicates the amount of unique fingerprinting features, but now the value is measured for each script. Next, we evaluated the scripts from four different points of view: the host domain, filename, script score, and the fingerprinting signature. We separated the script filename and host domain but also removed irrelevant paths and cache busters (e.g. script.js?v=123). In addition, we regrouped all scripts that have the same name but a different score, because they are most likely not the same scripts. Then we calculated a fingerprinting signature for each script. This signature represents the concatenation of all features that have been called in their order of occurrence, e.g., `UserAgent;Geolocation;Memory;CPU;CPU;...` With the help of this signature, we can match all scripts that have an equivalent behavior but have obfuscated filenames and source code or scripts that are bundled with other scripts. If a scripts randomize its function calls it can still be grouped based on the host domain, filename and script score.

Results: On the 10k most popular pages we found 72,457 scripts that are hosted on 6,896 unique domains. Scripts that do not use any JS function tracked by FPMON are not included. The large majority of scripts scored very low. One third (33.6%), use only a single monitored feature, while 97.6% of the scripts use 10 or fewer features. Hence, the absolute majority of scripts does not use many of the tracked features. We found 2,769 scripts to score at least 25%; 291 score at least 50%; and only 93 scripts reach a score of 75% and above. The data clearly shows that the majority of aggressive fingerprinting attempts are caused by less than 1% of the scripts on the 10k most popular websites. With respect to our baseline (panopticlick.eff.org), only around 300 scripts score similar or higher as the tool that can identify users with high probability. Moreover, we see that the script scores are in general lower than the page scores used in the previous studies. Since page scores are based on the concurrence of multiple scripts, it is likely that many of the high page scores are caused by multiple fingerprinting scripts that run concurrently.

Network	Score	Size	Data Sink	Examples of domain affected
Moatads	80%	58	3rd-party	breitbart.com, wsj.com, westernjournal.com, motor1.com, inquirer.net, nypost.com, ...
Sift	50%	45	3rd-party	udemy.com, scribd.com, patreon.com, kickstarter.com, wayfair.com, flickr.com, ...
Lalaping	88%	17	3rd-party	clipconverter.cc, shahid4u.net, swatchseries.to, o2tvseries.com, maxseries.tv, ...
Datadome	50%	16	3rd-party	nytimes.com, hepsiburada.com, leboncoin.fr, encuentra24.com, fnac.com, oui.sncf, ...
Adform	48%	31	3rd-party	freepik.com, coursehero.com, freepik.es, idnes.cz, tim.it, worldoftanks.eu, ...
Akamai	65%	232	1st party	adobe.com, rakuten.co.jp, foxnews.com, hulu.com, tokopedia.com, ikea.com, ...
fingerprint.js	48%	64	1st party	zhihu.com, agoda.com, olx.com.br, coinmarketcap.com, baixing.com, fmovies.to, ...
Google	20%	1343	3rd-party	reddit.com, okezone.com, twitch.tv, ebay.com, tribunnews.com, nytimes.com, ...

TABLE IV: The most prevalent script distributors with fingerprinting score and network size found with FPMON

Fingerprinting networks: We evaluated the scripts in our data set for matching signatures and identified 383 networks of different sizes. To reduce the results, we filtered the data by removing very small networks (size $<$ 10) and by manually merging those networks that obviously belong to the same entity, e.g., siftscience.com and sift.com, etc. Due to the limited space, Table IV shows only the most prevalent networks that we identified and analyzed more closely.

Most harmful networks: We find Sift [32] and Moat [27] to be the two most threatening networks due to their high number of fingerprinting features extracted and their relatively large distribution. Moat, owned by Oracle, is an ad-analytics platform. On their client's websites, they collect large amounts of user device data that are then sent to the Oracle network. Their scripts used 80% of the fingerprinting techniques monitored by us, 12 of those are labeled aggressive. Sift, as mentioned in the New York Times in 2019 [17], collects and builds reputation profiles of every Internet user. Overall, their scripts reach a score of 50% with 6 aggressive features and hence score similar to our baseline. Furthermore, we also found various smaller networks, e.g. created by companies like DataDome and Adform. DataDome scores 50% with 11 aggressive features and is present on 16 websites. Adform scores 48% with 4 aggressive features but affects almost twice as many websites. In all cases, the user data is collected on the client's website and sent to the network of the 3rd-party script provider.

Less harmful networks: We also found various networks that we believe to be less harmful. One of those is the Akamai network, which seems to be part of their bot detection service. Its distribution is surprisingly large and covers 232 websites, 4-5$\times$ more than Sift and Moat. Based on our analysis, the Akamai script appears to send the collected data diretcly to the website owner and not to Akamai itself, which might indicate benign behavior. Furthermore, we found the extremely large network of Google and its subsidiary DoubleClick with at least 1,343 websites. However, despite its huge distribution, they do not extract as much data as the other networks. Their scripts only score around 20% (with 2 aggressive features), not even one-third of what other services extract.

Miscellaneous networks: In our data set there are various other networks that share a common fingerprinting behavior. For example 'Lalaping', is a network of (illegal) streaming websites that share a common fingerprinting signature with a score of 88%. The script includes 13 aggressive features and is present on 17 websites within the 10k most popular pages. Likewise, some smaller networks are formed by other organizations that collect user data in the same way across all their brands and subsidiaries. Related to this are the 64 websites that include a version of fingerprintjs.com. In most cases the data is sent to the visited websites (1st party), however, we are unaware of why all this data is collected. One reason could be that contents are tailored based on the user profiles [20].

Key takeaways: Using FPMON, we were able to chart the landscape of fingerprinting networks and find it to be diverse and multi-dimensional. Many fingerprinting scripts are part of specific online services that ultimately collect vast amounts of user data. While Oracle's Moat and Sift have already cast a wide and threatening network of fingerprinting scripts, we can observe that smaller organizations are following their lead.

VII. Discussion

Having unraveled some of the key characteristics of JavaScript fingerprinting in the wild using FPMON, we now discuss the broader implications of our findings as well as highlight the limitations of our approach.

A widespread and hidden threat: In general, we noticed that fingerprinting scripts are designed to be stealthy: i) they do not interact with the user; ii) they do not ask for permission; and iii) they are explicitly not shown to the user (no GUI, no console log). They collect the device features, generate a fingerprint, and send it away within milliseconds, often before the page is even fully visible to the user. Moreover, a few scripts are even loaded and removed dynamically from the page (DOM) and others conceal the data transmission from the user using web sockets or ciphers. Ultimately, in many cases (e.g., the Moat and Sift networks) the amount of collected user device data is so extensive, that user identification is highly possible. This practice of concealed data collection heavily subverts current regulations on user's privacy such as the GDPR [13, 15]. To our knowledge, FPMON is the first tool that comprehensibly and reliably exposes this threat.

The need for better countermeasures: In Section VI-A we demonstrated that popular countermeasures are inadequate against JS fingerprinting, especially when the fingerprinting scripts are bundled within the scripts of the (1st party) website (as observed in Section VI-D). Hence, the blacklisting approach, as used in Firefox, DuckDuckGo, and Privacy Badger, cannot protect a user from fingerprinting scripts sufficiently. By reducing and unifying the JS interfaces without breaking functionality, as implemented by Apple, appears to be a more effective and sustainable solution against the problem. We hope, that FPMON and our findings will help to better understand the problem and build more effective protections against fingerprinting.

True intention of fingerprinting on websites: In Section VI-D we used FPMON to identify some of the networks that distribute fingerprinting scripts. None of the networks who reached a high score are present on a sufficiently large number of pages to reliably track users across the Internet. However, some organizations are on the edge of becoming a real threat to Internet users. Their network sizes might be comparatively small at the moment (typically $\leq$ 1% of the 10k most popular websites), but they often include high-profile pages and hence can analyze millions of users every day. Based on this evaluation, we also question the capacity in which owners know the practices and true power of the 3rd-party (fingerprinting) services used on their websites. For some of these networks, fingerprinting seem to be part of the tools that are used by the website administrator to maintain their services (e.g. bot detection, analytics, security). For example, archive.org, which has almost no fingerprinting activity (7%), however, their donation page scores 90% because of a single 3rd-party fingerprinting script. On the other side, nytimes.com scores usually 60% across their website, but deliberately disables all data collection on their dedicated whistleblowing subpage (0%).

Script vs Page Score: Another observation in our studies is the discrepancy between script and page scores in Section VI-C and VI-D. Although one could rate the webpage based on the highest script score, we believe that a comprehensive page score is more effective at judging data extraction caused by fragmented, dynamically loaded, and obfuscated scripts. For example, 41.7% of the 10k most popular websites include more than one script from a single 3rd-party domain. This kind of fragmentation can easily distort a script based score.

Performance: We evaluated potential slowdowns to page load times caused by FPMON and found the impact to be negligible. Under optimal test conditions, e.g. when there is no network transmission and all the monitored features are called, the overall slowdown is $\leq$ 20ms ($\leq$ 5% of the page load time). Under real-world conditions, e.g. when loading a webpage from a remote host with high fingerprinting activity such as metacafe.com, it took $4.760 \pm 200ms$ on average (10 measurements) to finish. Compared to the max. 20ms overhead, FPMON only adds under 0.5% to the total execution time under real-world conditions. Hence, we conclude that the user cannot perceive the tiny increase in execution time since network delays and rendering time are much larger than any slowdown caused by FPMON.

Limitations: In general, our findings suffer from the underlying problem that we cannot ascertain if a feature has been used for fingerprinting or not. For this reason, we chose a quantitative measure that covers all the components that are most typically used for fingerprinting. From our point of view, it is highly unlikely that a webpage uses all those features at once in a benign way. This is especially true for aggressive features such as CPU and memory information or the specific Canvas and Audio operations. In additions, user device fingerprinting can also be done without JavaScript, e.g. via Plugins [9], CSS [37] or with HTTP and TCP/IP data [33]. These techniques are beyond the scope of FPMON.

VIII. Related Work

In the following, we summarize related work in the domain of web browser fingerprinting. In 2009, Mayer [23] was able to identify 96% of 1,328 web clients, by hashing the concatenated contents of a set of four JavaScript objects [23]. Later on, Eckersley introduced the Panopticlick project [10]. In this experimental study, the team analyzed nearly half a million browsers with an extended set of features including the list of fonts, timezone, and various HTTP headers. This technique could identify 94.2% of the tracked devices. In addition, they published a novel way to extract the list of supported fonts by measuring the size of the rendered text and described this as one of the most accurate ways for device identification.

A few years later, mainly two studies have started to analyze the large scale adoption of fingerprinting techniques that are most closely related to FPMON. In 2013, FPDetective [2] introduced a crawler framework to study the use of font fingerprinting in the wild. The framework consists of an automated browser, a network proxy, and a flash decompiler. Using predetermined regular expressions, FPDetective can find the presence of 19 different font fingerprinters for JavaScript and Flash and concluded that nearly 1.5% of the Alexa 10k popular websites track users with font fingerprinting. Our analysis in Section VI-C shows that this has grown by an order of magnitude since then. In addition, their evaluation revealed that former protections (DoNotTrack, Mozilla Firegloves, and the Tor browser) can be bypassed in various ways and can make the user even more identifiable. Compared to FPDetective, FPMON adopts a broader approach, i.e., we detect the combined use of the most typically used fingerprinting techniques. Similar to FPDetective, we found the existing countermeasures to be ineffective against JS fingerprinting.

One year later, Acar et al. [1] analyzed the top 100k webpages and concluded that around 5.5% of the evaluated websites use canvas fingerprinting. To detect the canvas fingerprinting they modified the Firefox source code to log certain methods that are executed when a webpage uses the Canvas API. FPMON follows the same approach but does this for a comprehensive set of fingerprinting features and in a browser-independent way that does not require to modify the browser source code. In their study, Acar et al. discovered that the majority of the scripts (95%) belong to the same provider: addthis.com. As we have seen in our study (Section VI-A), their research probably had great impact, since addthis.com was the only provider found that respects user consent nowadays.

In 2016, Lerner et al. have published an archaeological study of web tracking from 1996 to 2016 [21]. The researchers created a tool called *TrackingExcavator* to make a longitudinal measurement though the Internet Archive's Wayback Machine. While their work focus on third-party tracking via cookies, they also measured the growing adoption of 37 fingerprinting API calls on the 500 most popular websites listed by Alexa. In comparison to FPMON, their interception approach is similar to ours, but the total function set is much smaller and the data shows almost no quantitative effects. Until 2016, almost half of the pages did not used more than 4 functions and less than 20 pages used 16 or more JS functions. Nevertheless, the study shows the historic adoption of JS functions and tracking networks.

In conclusion, several researchers have focused on specific functions, e.g. Font or Canvas fingerprinting, and typically with small function sets. With FPMON we take a broader perspective on the problem, i.e., we quantify the combined use of functions that are most typically used for fingerprinting to identify even unknown fingerprinting scripts.

IX. Conclusion

This paper was motivated by the increasing number of privacy violations posed by websites that apply JavaScript fingerprinting on their users. To that end, we conducted a systematic analysis of various popular fingerprinting tools to obtain an accurate understanding of the fingerprinting ecosystem. We found that JS fingerprinting is often well-hidden in the background and is usually done without user consent. Based on our classification and rating of JavaScript functions that are closely related to fingerprinting, we designed and developed FPMON, a light-weight and comprehensive fingerprinting monitor that can measure and rate JS fingerprinting on any given website in real-time.

Our evaluations using FPMON on real websites and with major browsers revealed the following. Several websites collect sprawling amounts of user data regardless of privacy regulations. Moreover, current countermeasures can not sufficiently protect users. The most practical and effective solution to thwart JS fingerprinting seems to be the reduction and unification of JS interfaces, as present in the Safari browser. In our study of the Alexa top 10k websites, we found that i) fingerprinting has grown tremendously in the past years (by an order of magnitude for font fingerprinting); ii) nearly one in five websites aggressively collects user data via JS fingerprinting; and iii) half of the identified JS features that are closely related to fingerprinting are unlikely used in benign applications. Finally, using FPMON we identified the diverse networks that foster the use of JS fingerprinting. Some of these networks openly admit to profile users, others integrate fingerprinting into complex services for website owners such as bot detection, analytics, and security tools, that collect huge amounts of user data on their client's websites. On many affected websites, the amount of data collected is so extensive, that precise user identification becomes very likely with regard to previous research [10, 35]. We believe that harvesting such vast amounts for device data without user consent does not justify its purpose and poses a damaging threat to web user privacy [15, 4, 29].

We hope this paper and FPMON helps to empower web users to uncover how and where their data is collected while browsing the web. Beyond that, we find FPMON to have three more contributions to the web ecosystem. First, using FPMON, website owners can scrutinize 3rd-party components for concealed fingerprinting behavior to improve the privacy of their websites. Second, browser vendors can test when and why their protections fail and improve their solutions. Third, FPMON can be integrated continuously, e.g. into search engines: websites with poor privacy behavior can be ranked lower in the search results to counter the adoption of JS fingerprinting and protect user's privacy on a large scale.

References

[1] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. "The web never forgets: Persistent tracking mechanisms in the wild". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 2014, pp. 674–689.

[2] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. "FPDetective: dusting the web for fingerprinters". In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 2013, pp. 1129–1140.

[3] Alexa Internet Inc. *The top 10,000 sites on the web*. https://www.alexa.com/topsites.

[4] Apple Inc. *Safari Privacy Overview. Learn how the Safari web browser protects your privacy.* https://www.apple.com/safari/docs/Safari_White_Paper_Nov_2019.pdf. 2019.

[5] BrowserLeaks. *A gallery of browser fingerprinting demos*. https://browserleaks.com/.

[6] California State Legislature. *California Consumer Privacy Act (CCPA), Assembly Bill No. 375*. https://leginfo.legislature.ca.gov/. 2018.

[7] Yinzhi Cao, Song Li, and Erik Wijmans. "(Cross-) Browser Fingerprinting via OS and Hardware Level Features." In: *NDSS*. 2017.

[8] DuckDuckGo. *DuckDuckGo Privacy Essentials*. https://duckduckgo.com/app.

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

[10] EFF. *Panopticlick 3.0 - Is your browser safe against tracking?* https://panopticlick.eff.org/.

[11] Peter Eckersley for the Electronic Frontier Foundation. *A Primer on Information Theory and Privacy*. https://www.eff.org/deeplinks/2010/01/primer-information-theory-and-privacy.

[12] Steven Englehardt and Arvind Narayanan. "Online tracking: A 1-million-site measurement and analysis". In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 1388–1401.

[13] European Parliament and Council of the European Union. *The EU General Data Protection Regulation (GDPR)*. https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[14] Electronic Frontier Foundation. *Privacy Badger*. https://privacybadger.org/.

[15] Electronic Frontier Foundation. *The GDPR and Browser Fingerprinting: How It Changes the Game for the Sneakiest Web Trackers*. https://www.eff.org/de/deeplinks/2018/06/gdpr-and-browser-fingerprinting-how-it-changes-game-sneakiest-web-trackers.

[16] Google LLC. *Chrome DevTools - web developer tools*. https://developers.google.com/web/.

[17] Kashmir Hill. *I Got Access to My Secret Consumer Score. Now You Can Get Yours, Too.* https://www.nytimes.com/2019/11/04/business/secret-consumer-score-access.html.

[18] Hotjar. *The fast & visual way to understand your users*. https://www.hotjar.com/.

[19] Troy Hunt. *Have i been pwned? Check if you have an account that has been compromised in a data breach*. https://haveibeenpwned.com/.

[20] Thomas Hupperich, Dennis Tatang, Nicolai Wilkop, and Thorsten Holz. "An empirical study on online price differentiation". In: *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. 2018, pp. 76–83.

[21] Adam Lerner, Anna Kornfeld Simpson, Tadayoshi Kohno, and Franziska Roesner. "Internet jones and the raiders of the lost trackers: An archaeological study of web tracking from 1996 to 2016". In: *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 2016.

[22] MaxMind, Inc. *MaxMind's minFraud Score, minFraud Insights, and minFraud Factors API*. https://dev.maxmind.com/minfraud/.

[23] Jonathan R Mayer. "Any person... a pamphleteer": Internet Anonymity in the Age of Web 2.0". In: *Undergraduate Senior Thesis, Princeton University* (2009).

[24] Keaton Mowery and Hovav Shacham. "Pixel perfect: Fingerprinting canvas in HTML5". In: *Proceedings of W2SP* (2012), pp. 1–12.

[25] Mozilla. *Enhanced Tracking Protection in Firefox for desktop*. https://support.mozilla.org/en-US/kb/enhanced-tracking-protection-firefox-desktop.

[26] Mozilla. *Protections Against Fingerprinting and Cryptocurrency Mining Available in Firefox Nightly and Beta*. blog.mozilla.org/futurereleases/2019/04/09/.

[27] Oracle Corporation. *Moat Analytics: Real-time, multi-platform, and actionable marketing analytics.* https://moat.com/.

[28] PortSwigger Ltd. *Burp Suite, Web vulnerability scanner*. https://portswigger.net/burp.

[29] Prabhakar Raghavan. *Raising the bar on transparency, choice and control in digital advertising*. https://blog.google/products/ads/transparency-choice-and-control-digital-advertising/.

[30] Princeton Web Transparency & Accountability Project. *AudioContext Fingerprint Test Page*. https://audiofingerprint.openwpm.com/.

[31] Michael Schwarz, Florian Lackner, and Daniel Gruss. "JavaScript Template Attacks: Automatically Inferring Host Information for Targeted Exploits." In: *NDSS*. 2019.

[32] Sift Science, Inc. *Digital Trust & Safety: Go beyond fraud prevention with Sift*. https://sift.com/.

[33] Matthew Smart, G Robert Malan, and Farnam Jahanian. "Defeating TCP/IP Stack Fingerprinting." In: *Usenix Security Symposium*. 2000.

[34] Peter Snyder, Cynthia Taylor, and Chris Kanich. "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*. 2017, pp. 179–194.

[35] SPIRALS Research Group from University of Lille. *Browser Fingerprinting Test*. https://amiunique.org/.

[36] Oleksii Starov and Nick Nikiforakis. "Xhound: Quantifying the fingerprintability of browser extensions". In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017, pp. 941–956.

[37] Naoki Takei, Takamichi Saito, Ko Takasu, and Tomotaka Yamada. "Web browser fingerprinting using only cascading style sheets". In: *2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*. IEEE. 2015, pp. 57–63.

[38] TransUnion LLC. *We make it safer for you to do business online.* https://www.iovation.com/.

[39] Erik Trickel, Oleksii Starov, Alexandros Kapravelos, Nick Nikiforakis, and Adam Doupé. "Everyone is different: client-side diversification for defending against extension fingerprinting". In: *28th USENIX Security Symposium (USENIX Security 19)*. 2019, pp. 1679–1696.

[40] Valentin V. *Open source browser fingerprinting library*. https://github.com/Valve/fingerprintjs2.

[41] Antoine Vastel, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. "FP-scanner: the privacy implications of browser fingerprint inconsistencies". In: *27th USENIX Security Symposium (USENIX Security 18)*. 2018, pp. 135–150.

[42] Antoine Vastel, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. "FP-STALKER: Tracking browser fingerprint evolutions". In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018.

[43] Shujiang Wu, Song Li, Yinzhi Cao, and Ningfei Wang. "Rendered Private: Making GLSL Execution Uniform to Prevent WebGL-based Browser Fingerprinting". In: *28th USENIX Security Symposium (USENIX Security 19)*. 2019, pp. 1645–1660.

[44] Yahoo Developer Network. *How Many Users Have JavaScript Disabled*. searchenginepeople.com/stats-no-javascript.html.