Too Little Too Late: Can We Control Browser Fingerprinting?

Nasser Mohammed Al-Fannah
Information Security Group
Royal Holloway, University of London
nasser@alfannah.com

Chris J Mitchell
Information Security Group
Royal Holloway, University of London
me@chrismitchell.net

Abstract

Browser fingerprinting is increasingly being used for online tracking of users, and, unlike the use of cookies, is almost impossible for users to control. This has a major negative impact on online privacy. Despite the availability of a range of fingerprinting countermeasures as well as some limited attempts by browser vendors to curb its effectiveness, it remains largely uncontrolled. Third-party countermeasures have inherit limitations and many browser vendors do not appear to have made significant efforts to control it. This paper provides the first comprehensive and structured discussion of measures to limit or control browser fingerprinting, covering both user-based and browser-based techniques. It also discusses the limitations of these measures and the need for browser vendor support in controlling fingerprinting. Further, a somewhat counterintuitive possible new browser identifier is proposed which could make cookies and fingerprint-based tracking redundant; the need for, and possible effect of, this feature is discussed.

1 Introduction

Browser fingerprinting appears to have become a somewhat commonly used technique for online tracking [3], i.e. linking multiple visits by a single browser to the same website, and/or linking individual visits by a browser to multiple sites. For many years, both types of tracking have been made possible through the use of cookies, where third party tracking sites can link multiple site visits through inclusion of their content on cooperating sites. However, fingerprinting is far more persistent than cookie-based tracking, virtually uncontrollable by users and non-trivial to detect. Moreover, fingerprinting can be used to create *supercookies*, where if a tracking cookie is deleted from a user platform, it can be regenerated if the same browser is detected via fingerprinting [7].

The fact that tracking can so readily be performed using browser fingerprinting is potentially a major threat to the privacy of web users, and as noted above it is one over which users currently have no control. Whilst there are uses of browser fingerprinting not directly relating to tracking, the lack of user control combined with the serious privacy threat suggests that means of limiting its effectiveness, i.e. what we refer to here as *fingerprinting countermeasures*, are of potentially huge importance, motivating this paper.

Fingerprinting countermeasures can be divided into two categories, depending on whether they are directly implementable by the user regardless of the browser or whether they require support from the browser vendor. In the remainder of this paper we provide a comprehensive and systematic review of possible fingerprinting countermeasures. This is significant for a number of reasons. First, some of these techniques, whilst apparently known, have not previously been described in the academic literature. Second, this review enables us to compare their effectiveness (and their limitations) and also consider how best such countermeasures could be implemented, from the perspectives of both the user and the browser vendor. Third, it enables us to identify areas where further research is urgently needed.

Finally, given that fingerprint-based tracking is so privacy intrusive (and uncontrollable), we also consider a way in which the major fingerprinters might be encouraged to abandon the practice. We, possibly controversially, propose that browsers should support a new type of website-accessible identifier, referred to as a *Unique Browser Identifier*, which would enable a level of user-controllable tracking without involving the collection of other user and browser data. This could make browser vendors willing to change behaviour to make fingerprinting difficult, leading to its use becoming redundant and (potentially) prevented. The possible operation of this identifier, and its advantages and disadvantages, are discussed.

The remainder of the paper is organized as follows. Section 2 provides a brief introduction to browser fingerprinting. In Section 3, a general overview of approaches to limiting fingerprinting is provided. Sections 4 and 5 provide detailed descriptions of all known user-based and browser-based anti-fingerprinting measures, respectively. In Section 6, we discuss a browser identifier-based proposal that aims at making browser fingerprinting redundant. Building on the previous sections, in Section 7 we review the degree to which browser fingerprinting can be controlled using current technology and consider ways in which greater control can be exercised in the future.

Acknowledgements the authors would like to thank the anonymous referees for their helpful comments and suggestions.

2 Browser Fingerprinting

Browser fingerprinting, as first described by Eckersley [7], is a technique that allows web servers to uniquely identify user devices by examining information retrievable from a browser, where this collection of information is unique for most instances. There are various possible uses for fingerprinting, but one of the most widely discussed (and controversial) is *online tracking* (i.e. enabling web servers to link multiple interactions with the same platform). Since Eckersley first described it, the range and richness of information retrievable from a browser that is usable for fingerprinting has substantially increased, as has real-world deployment of fingerprinting by websites [3, 20].

Browser fingerprinting can be performed by active or passive means [6]. *Passive* fingerprinting depends entirely on information retrievable through regular HTTP requests such as the HTTP header field *user agent*[1], whereas *active* fingerprinting involves the use of scripts to retrieve further information about the browser and its configuration, such as the set of installed fonts.

As has been widely discussed, for example by Eckersley [7], Narayanan and Reisman [23], and Perry [30], there are a number of reasons why browser fingerprinting represents a more significant threat to user privacy than cookies.

- Typically there is no simple way to determine for certain whether a website is deploying any of the various browser fingerprinting techniques.
- A user can limit the tracking power of cookies in a number of ways, e.g. by regularly deleting cookies or blocking them altogether (as supported by most browsers), but there are no comparable, easily configured, means of limiting fingerprinting.
- Unlike cookies, browser fingerprinting is not dependent on a single explicit feature of HTTP. Fingerprinting rather relies on many techniques to collect various information about the properties and configuration of the browser and its host platform. Any of this information has the potential to be used for fingerprinting.

3 Limiting Browser Fingerprinting

3.1 General Approaches

Most techniques aimed at limiting the effectiveness of fingerprinting either involve user-enabled options such as installing extensions, or operate via

[1]This header field contains information related to the browser and its host, such as the browser version, operating system version and screen resolution [12].

browsers that incorporate anti-fingerprinting features. We discuss these two general classes of countermeasures in greater detail in Sections 4 and 5 below.

A number of authors have proposed extensions that could help counter fingerprinting. Examples include *FP-Block* [36], *Blink* [19] and *Fingerprint-Alert* [3]. There are also widely discussed extensions of this type that do not seem to have any corresponding published research, such as *CanvasBlocker*[2] and *Stop Fingerprinting*[3]. Luangmaneerote et al. [21] surveyed several of the more widely discussed fingerprinting countermeasures. They concluded that no single countermeasure can protect against all known fingerprinting methods, such countermeasures tend to negatively affect the user experience, and are likely to fail to block newly deployed fingerprinting methods. In this paper, we do not attempt to evaluate or enumerate individual anti-fingerprinting extensions; instead our goal is to consider the possible general approaches and in each case examine its effectiveness.

As far as browser-incorporated anti-fingerprinting techniques are concerned, a number of documents provide recommendations and guidelines to browser vendors aimed at limiting the effectiveness of fingerprinting, including RFC 6973 [11], a W3C Note [6], Eckersley [7], and Nikiforakis et al. [26]. However, the detailed technical aspects of such recommendations are outside the scope of this paper, which is intended to provide a roadmap for policymakers, developers, browser vendors and interested general web users who wish to help control browser fingerprinting.

3.2 Challenges

A number of proposed browser fingerprinting countermeasures involve adding features to the browser, typically in the form of a browser extension. However, approaches of this type have serious limitations [21] and might even lead to effects opposite to those intended (see 4.3). By contrast, and as discussed in greater detail in Section 5, browser vendors could potentially control fingerprinting very effectively by making modifications to the way browsers operate. To help substantiate these claims, we next consider some of the main challenges in controlling browser fingerprinting.

3.2.1 Misuse of Browser Features

One major challenge in controlling fingerprinting is that information useful for fingerprinting can be obtained from "regular" web interactions, including website-originated browser scripts that access standardized APIs[4]. That is, preventing fingerprinting might necessitate stopping, or restricting, such

[2]`https://addons.mozilla.org/en/firefox/addon/canvasblocker/`

[3]`https://addons.mozilla.org/en/firefox/addon/stop-fingerprinting`

[4]A web API, i.e. a web Application Programming Interface, is a set of functions or methods that can be used to access certain functionality of web browsers.

interactions, scripts and the APIs they access, which will almost certainly damage the user's browsing experience. For example, *canvas fingerprinting*[5] makes use of the Canvas API [5], and simply blocking this API would result in browsers being unable to render images that could be critical to use of a web page.

3.2.2 Detection

Unlike tracking via cookies, that can be detected by the presence of cookies stored on user device, there are no unambiguous methods of detecting browser fingerprinting [26]. Moreover, it can be performed passively and is thus virtually undetectable (except, perhaps, by second order effects, such as receipt of targeted advertising). As described by Narayanan et al. [23] and Perry et al. [31], active fingerprinting can also be very hard to detect, as it can take advantage of almost any existing web API and almost certainly any future new APIs[6].

4 User-based Countermeasures

We now describe and analyse a variety of ways in which users can reduce the effectiveness of browser fingerprinting. We also consider the main challenges to user-based approaches, in particular observing that none of these techniques prevent fingerprinting — indeed, some may even make it more effective.

4.1 Browser Choice and Configuration

Browsers vary in their susceptibility to fingerprinting [2]. These variations arise for a variety of reasons, including that some browsers, such as Firefox, have (possibly user-selectable) features that are designed to help resist fingerprinting. Moreover, as described by Boda et al. [4], some browser configuration options, such as disabling JavaScript, can affect fingerprinting. That is, a user's choice of browser and configuration options can change the effectiveness of fingerprinting. Finally, selecting a browser and version that is used by many users is also likely to help in making the browser a little less fingerprintable[7].

Before proceeding it is important to mention the Tor browser[8], which is specifically designed to be privacy-protecting; hence selecting Tor could be seen as a potentially effective user-based countermeasure. Unlike several

[5]A browser fingerprinting technique that depends on exploiting the Canvas API [22].

[6]Unless new APIs are developed with built-in resistance to fingerprinting.

[7]`https://panopticlick.eff.org/self-defense`

[8]The Tor browser is a modified version of Firefox browser that has enhanced security and privacy features, `https://www.torproject.org/projects/torbrowser`

widely-used browsers, the Tor browser includes by default a range of features intended to counter fingerprinting. These include using a fixed set of system colours, disabling plugins by default, limiting the number of fonts a document is allowed to load, and disallowing read access to canvas-rendered images unless express permission is granted by the user[9].

However, use of the Tor browser has a number of serious practical drawbacks including a seriously compromised browsing experience [16], one aspect of which is a slow browsing speed; there are also relatively few users (less than 1% of web users employ the Tor browser[10]) which might itself make fingerprinting possible because of the fingerprintability paradox (see 4.3). Tor also possesses other serious usability issues: it breaks some websites, some websites opt to block Tor clients, and changes in IP address negatively affect the localized web browsing experience. Another example of a usability issue arises when Tor advises a user against maximizing the browser window via an in-browser notification. This is intended to keep the window at the default size and thereby the same size as other Tor users, hence preventing websites from learning the device's display size. Whilst this reduces browser fingerprintability, it will clearly have a negative effect on the user experience.

More generally, making attributes the same across multiple browser instances and hence reducing their uniqueness would make fingerprinting more difficult. However, it is also likely to damage the user experience by preventing the website tailoring its site to match the characteristics of the user device [31]. These serious disadvantages mean that the Tor browser is unlikely ever to be widely adopted, and hence cannot be seen as generally applicable a means of controlling fingerprinting.

4.2 Browser Extensions

Apart from the fundamental choice of browser, the main option for users wishing to limit the effectiveness of browser fingerprinting is to install one or more special-purpose browser extensions. As noted in Section 3.1, a number of such extensions exist, and we now consider these extensions in greater detail.

We were unable to find any detailed and generally applicable evidence regarding the relative effectiveness of the existing extensions; however they all seem to share common weaknesses. The limited effectiveness of some individual extensions has been evaluated in controlled environments, including by Nikiforakis et al. [27] and Luangmaneerote et al. [21]. It seems unlikely

[9]`https://www.torproject.org/projects/torbrowser/design` [accessed 13/02/2019].

[10]Estimated by comparing the number of Tor browser clients during January 2019 as reported on `https://metrics.torproject.org` with the total number of web users reported on `http://www.internetlivestats.com/internet-users`.

that a single extension is able to completely prevent fingerprinting, given the multiplicity of fingerprinting approaches, potentially including methods not in the public domain. Furthermore, there is no known method of learning whether extensions that counter certain fingerprinting techniques are in fact able to prevent real-world fingerprinting. This is especially apparent given that it is not always possible to detect when fingerprinting is occurring in the first place [1].

We also observe that some extensions not specifically designed for the purpose can nonetheless reduce the effectiveness of fingerprinting, as a by-product of their intended functionality. One such example is NoScript[11], that controls which scripts deployed by a visited website are allowed to run on the browser; depending on its configuration it might prevent execution of scripts used for browser fingerprinting. However, in this study we only consider purpose-built anti-fingerprinting extensions.

We next briefly review the three main techniques employed by the anti-fingerprinting browser extensions of which we are aware. We consider the limitations they all share in Section 4.3 below.

- **Script Blocking**: this works by blocking suspected fingerprinting scripts. One example of an extension adopting this approach is *PrivacyBadger*[12], developed by the Electronic Frontier Foundation. It detects canvas fingerprinting and prevents third-party scripts that deploy it from executing.

- **Attribute Spoofing**: extensions that use this technique attempt to prevent fingerprinting by constantly spoofing browser/platform attributes. Examples of extensions using this approach include: *PriVaricator* (due to Nikiforakis et al. [25]), *FPRrandom* (due to Laperdrix et al. [18]), *FPGuard* (due to FaizKhademi et al. [9]) and an unnamed extension due to Fiore et al. [13]. *FP-Block*, due to Torres et al. [36], also fabricates some browser attributes, but it also blocks some scripts. That is, it employs both *script blocking* and *attribute spoofing.*

- **Data Blocking**: this involves blocking the retrieval of attributes that might be used for fingerprinting from the browser. This approach is used by *FingerprintAlert*[13].

4.3 Limitations

The main disadvantage of user-based countermeasures is that they all depend, to a limited extent, on manipulating or blocking data sent by the

[11]`https://noscript.net`

[12]`https://www.eff.org/privacybadger`

[13]`https://addons.mozilla.org/en/firefox/addon/fingerprintalert/`

browser to remote web servers. This gives rise to several limitations, which we now discuss.

- **Limitation of Browser Choice:** although browsers vary in their fingerprintability, they still exhibit a large number of fingerprintable attributes. Perhaps this is less true for privacy-hardened browsers such the Tor browser but it comes at the cost of a compromised browsing experience.

- **Limitation of Extensions:** browser extensions can customize the attributes and behaviour of a browser. However, their ability to modify behaviour is by definition limited to what browser vendors allow. This means that browser extensions cannot replace vendor-implemented measures that enhance the privacy properties of their browsers.

- **Compromised Browsing Experience**: many browser-based countermeasures compromise the browsing experience in some way [21]. This is especially true when such countermeasures block certain scripts or spoof properties that may be important for the functionality of a visited website. There is also the possibility of a false positive, i.e. an incorrectly detected fingerprinting attempt, breaking some "innocent" websites.

- **Fingerprintability Paradox**: this a phenomenon that has been widely discussed — see, for example, Eckersley [7] and Torres et al. [36]. The term captures the fact that measures taken to reduce browser fingerprintability can unintentionally create a new source of fingerprinting. A simple example arises where an anti-fingerprinting browser extension that is only installed in a small number of devices can be detected by a web server. That is, the presence of the extension is itself an attribute that can contribute to fingerprinting. This is a special case of what to refer to as *detection of defence*, where the deployment of a countermeasure can be detected and can be used to contribute to fingerprinting [26, 35]. This problem is especially significant if the number of users of the countermeasure is relatively small.

  A related but distinct issue arises from the deployment of a browser extension that spoofs browser attributes for anonymization purposes but as a result exhibits a set of browser characteristics that is unrealistic or rare. This behaviour can be used to make a browser more identifiable. Further, Vastel et al. [37] argue that some countermeasures potentially make browsers more fingerprintable since both spoofed and correct browser attributes can be discovered, e.g. using two different APIs. Finally, it has also been observed by Perry [30] that privacy-cautious users who opt to enable the *Do Not Track* (DNT)[14] option in

[14]DNT is a standard browser feature that sends a request to websites that the browser

their browsers increase their fingerprintability surface by doing so. In fact, the DNT option is no longer officially endorsed by the W3C [33]; while it is still supported by some browsers, Apple has announced[15] that version 12.1 of its Safari browser will drop support.

5 Browser-based Countermeasures

We next discuss the various countermeasures that could be implemented by browser vendors, as well as the associated challenges.

5.1 Reducing the Fingerprinting Surface

Having browsers exhibit similar information wherever possible would help limit fingerprinting [6, 11]. For example, the HTTP user agent header field is an important source of information for fingerprinting [14, 38] since as currently implemented it contains many browser and platform details, including full details of the browser version and, in some mobile browsers, the mobile phone model [2]. Restricting the information included to only what is vital for the functioning of websites would clearly help reduce its utility for fingerprinting, whilst not affecting its usefulness for tailoring website content.

Currently, the specifics of API implementation are typically left to browser vendors, which increases their usefulness for fingerprinting by enabling one browser to be distinguished from another through minor implementation differences [23]. However, if the standards included enough details to ensure these APIs are implemented in a way that would make browsers indistinguishable, then this would in turn limit fingerprinting effectiveness.

One of the methods used by some browsers (e.g. Firefox) to limit fingerprinting is attribute spoofing (as used in anti-fingerprinting extensions, cf. 4.2). However, as discussed in Section 4, attribute spoofing can seriously damage the user experience, and may also have a computational cost. Thus, whilst attribute spoofing may be necessary as a short-term expedient, in the longer term arranging for as much cross-browser attribute uniformity as possible is clearly a preferable approach [31]. As a result, it is likely that browsers that resort to spoofing are doing so as a temporary measure, as achieving behavioural uniformity would require, a currently absent, consensus amongst browser vendors.

In conclusion, and as recommended by RFC 6973 [11], it would be highly desirable for browsers to minimize the information content of browser-retrievable attributes to that needed to deliver the user experience, whilst

user does not wish to be tracked [33].

[15]`https://developer.apple.com/documentation/safari_release_notes/safari_12_1_release_notes` [accessed 13/02/2019]

also working to remove unnecessary differences in browser behaviour, including the order and presence of HTTP fields. Such changes could make a significant difference to the effectiveness of fingerprinting, with no obvious disadvantages in terms of the delivery of web content.

5.2 Context-based API access control

If browsers could be designed to control access to certain APIs based on the context, this would be very useful in controlling fingerprinting. For example, and as demonstrated by the *Fingerprintability* test web page[16], the client platform private IP address can be exposed via the WebRTC API. The specific feature of WebRTC that reveals the IP address is intended for video chat purposes. Currently, again as shown by *Fingerprintability*, this API can be accessed in many widely-used browsers regardless of whether or not video conferencing is taking place (or even without prompting the user). Hence (by some means) restricting access to the WebRTC API to video chat sites would clearly be beneficial in limiting fingerprinting.

There are many other examples of APIs whose use could be limited with similar benefits, such as access to processing and graphics hardware information through the WebGL API[17] in cases where it is not used to render any graphics. Analogously, since tracking is mostly performed by third parties, it would be very helpful if third-party scripts were prevented from accessing any API unless there is a clear reason for its use by a party other than the visited website [31].

5.3 Deprecate/Limit Unnecessary APIs

There is a need for the set of standardized APIs to be revisited and pruned where possible, since some APIs are apparently almost exclusively used by fingerprinters [31]. Of course, such APIs were not developed for fingerprinting purposes. Supporting this, Snyder et al. [34] have shown that many APIs are not utilized by any of top 100,000 visited websites. Such lightly used APIs create avoidable fingerprinting opportunities. Removing such APIs, or at least limiting their functionality, would therefore limit fingerprinting with minimal impact on the user experience. Two examples of such APIs are as follows.

- As noted by Olejnik et al. [29], the battery API is almost solely used for fingerprinting purposes. This API allows websites to detect the battery level of the client's device and optionally adjust website content to reduce battery-draining features if low battery levels are detected [17]. Safari never supported this API while Firefox did support it for a

[16]`https://fingerprintable.org/webrtcleaks`

[17]A web API that renders 2D and 3D graphics on supported browsers.

while but deprecated it in 2017 (possibly because of privacy issues). This move was followed by several other browser vendors. At the time of writing, Chrome is the only one of the top five browsers that still supports this API.

- An example of an API with features which seem primarily useful for fingerprinting is provided by the Canvas API, that enables a website to specify an image in code form (reducing data transfer requirements). Mowery and Shacham [22] showed that browsers and platforms vary in how they render canvas images. This is made valuable for fingerprinting by an API feature that allows a website to retrieve the canvas-rendered image[18]. Disabling this latter feature would remove the fingerprinting function without preventing use of the API for its intended purpose.

5.4 Alerts and Prompts

As noted by Doty [6] and Fette and Melnikov [11], it would enhance user control if users were made aware whenever a browser detects behaviour which suggests fingerprinting is being performed. In addition, it would also be helpful if users could be given the means to control such behaviour. Of course, as we have discussed above, reliably detecting fingerprinting is a hard problem. However, it might be possible to detect when a website is collecting information which is apparently unrelated to the information being served to the user. This could, perhaps, involve the use of machine learning techniques.

Examples of possible controls that could be given to users when a website is detected collecting fingerprintable data include:

- **blocking** the data collection;
- **anonymizing** the collectable data by spoofing or removing unique values;
- allowing users to **select** what data can be collected by a visited website.

Even if it is not possible to control the potential fingerprinting, it would help if users could be notified when such activity, e.g. involving the Canvas API, is detected. To a limited, and varying, extent this is implemented in both the Safari and Tor browsers. However, browser vendors need to be wary of *warning fatigue* [31], as over-frequent alerts might cause users to pay less attention to prompts and click on them without considering their content. Usefully, the number of times a user is prompted could be reduced if

[18]The API allows the retrieval of the binary pixel data of the rendered image by the visited website [22]

appropriate options were made available to users, such as enabling/disabling prompts, blacklisting, and automatically blocking certain third-party interactions. Moreover, prompts could be reduced if users were prompted only when suspicious behaviour is detected. An example of such suspicious behaviour would be a website that attempts to retrieve the canvas-rendered image while the image itself cannot be seen by the user because it is invisible or too small.

In a small investigation we found that none of the top four desktop browsers[19] (i.e. Chrome, Firefox, Internet Explorer and Edge) alert users when a visited website performs actions typical of fingerprinters, nor do they appear to incorporate any specific measures to prevent fingerprinting[20]. By contrast, as discussed in 4.1, the Tor browser protects against canvas fingerprinting by default as it prompts users before allowing the retrieval of canvas-rendered images by a visited website.

5.5 Reduction in API accuracy

Several studies (e.g. by Eckersley [7] and Olejnik at al. [28]) have suggested using a reduction in the accuracy and level of detail provided by a browser in order to help counter fingerprinting. As discussed in 5.1, reducing the level of detail in the HTTP header would significantly help in limiting fingerprinting. The reporting of constantly varying values, such as time, location and battery level could also be made less accurate to help prevent fingerprinting.

A further example of this type is provided by location information. Currently, all widely-used browsers prompt users to give permission if a website tries to access the location of the user. However, while the currently high level of accuracy obtainable (e.g. up to 4–5 metres) is likely to be necessary for applications such as satellite navigation, the need for such accuracy for most cases is arguable at best. Reducing the level of accuracy of the data provided, e.g. by enhancing the API to allow two or more levels of accuracy, could help limit fingerprinting. In addition to the prompt for user permission to access location information implemented by many browsers, the browser could also *warn* the user if a website is requesting highly-accurate location information.

5.6 Secure Data Handling

As reported in a previous study [3], potentially privacy-sensitive fingerprinting data is often retrieved from a browser in plaintext via HTTP. Browsers could usefully enforce the use of HTTPS for such information transfers,

[19]The list of the most widely-used browsers was retrieved from `https://www.netmarketshare.com/browser-market-share.aspx` [accessed 01/10/2019].

[20]Firefox has a small set of configurable anti-fingerprinting settings; however, these are only likely to be employed by technically aware users.

as advised in RFC 6973 [11]. Currently, Chrome is the only browser that forces websites to use HTTPS in order to access the user location through the Geolocation API[21]. However, this restriction is not extended to other APIs.

Currently, several browsers, e.g. Firefox, warn users if they are visiting a website not using HTTPS, and prompt users to deny transmission if they are about to submit information that does not use it. However, no warnings and almost no restrictions are in place if a script from a third-party website retrieves information via HTTP. This shortcoming clearly merits consideration by browser vendors.

5.7 Challenges

Perhaps the main challenge for implementing browser-based countermeasures is to make browsers, wherever possible, behave indistinguishably to websites while keeping retrievable information to a minimum. Achieving the necessary agreement amongst competing vendor providers is likely to be difficult without enforcement by regulatory or standard bodies.

Another obvious challenge is ensuring changes implemented by browser vendors have minimal negative impact on the browsing experience of users. Achieving this is non-trivial given that, as discussed earlier, changes could include restricting APIs as well as occasionally prompting users.

Finally, some browser vendors might not wish to limit fingerprinting given that their parent companies apparently depend on it for their own services, such as providing web analytics and personalized online advertising [3].

6 Making Browser Fingerprinting Unnecessary?

6.1 A Different Approach

As noted briefly above, whilst browser vendors are in a strong position to decide how effective browser fingerprinting is, some of the key players in this space, notably Google, may be unlikely to take steps to limit it since they also appear to play a major role in browser fingerprinting [3]. That is, there is clearly a desire by key browser vendors to be able to track user behaviour. Indeed, to some extent this is necessary to enable these vendors to continue to support “free” (and highly valued) services, such as web search.

In this respect, it might be argued that trying to restrict cookies has been counter-productive for user privacy; at least cookies are, to a high degree, user-controllable, i.e. users can delete all cookies from time to time and thereby refresh their online identity. Exerting usage control is much more difficult when browser fingerprinting is employed for tracking, since as

[21]`https://www.w3schools.com/html/html5_geolocation.asp` [accessed 18/02/2019]

we have argued it is far less controllable and far more privacy-damaging in that it retrieves a wide variety of information about a user's platform and browser configuration. That is, pressure to limit cookies may have encouraged trackers to adopt browser fingerprinting, with an associated worsening of end-user privacy protection.

Apparently, five years ago Google intended to replace the use of cookies with some form of browser-generated ID (confirmed by a Google official [15]). However, no further information was ever made available. This apparently abandoned proposal suggests a possible new, and apparently paradoxical, approach to limiting browser fingerprinting. That is, if trackers can be offered a means of tracking browsers that is less privacy-damaging than browser fingerprinting, then browser vendors might be more willing to countenance adopting measures to reduce the effectiveness of fingerprinting.

Currently, operating systems such as Windows and Android support an *advertising ID* [32]. This serves as a unique identifier for the device/user for locally installed programs/apps to serve personalized ads. This ID can be reset, meaning that all previous associations are removed. This suggests a similar scheme in which a novel user-controllable ID is accessible by websites through browsers. This ID (which we call the *Unique Browser Identifier* (UBI)) would be managed by the browser itself, rather than the host operating system, since it is intended for use solely by the browser.

The main reason to introduce a browser-managed UBI is to provide a replacement for both cookies and browser fingerprinting for the purposes of tracking, e.g. for the support of personalized advertising. That is, by providing a legitimate, simple and user-controllable method of enabling tracking, the use of cookies and browser fingerprinting could be made redundant. Simultaneously with the introduction of the UBI, measures also need to be put in place to prevent trackers using both the UBI and browser fingerprinting (hence damaging privacy even further). This could be achieved by regulation and/or standardization, as well as by implementing the recommendations given in Section 5. Apart from making fingerprinting for tracking redundant, the UBI could also replace other existing uses of browser fingerprinting. In particular it could serve as an additional layer of authentication.

6.2 Configuring Identifiers

To ensure that the introduction of the UBI gives the user the control that is currently lacking with browser fingerprinting, browsers will need to enable users to limit access to the UBI (and reset it). It could be advantageous for a browser to support more than one UBI, for example one for personalization (e.g. for personalized advertising) and another for authentication purposes (e.g. as a factor in multi-factor authentication). This would allow the access settings for the various UBIs to be separately configurable. For example, a user might choose to allow third parties to have access to a personalization

UBI, e.g. as used for personalized advertising, and might also choose to automatically reset this UBI at fixed intervals (e.g. monthly). In parallel, an authentication UBI might be configured to only be available under very restricted circumstances, e.g. only to the site visited by the user and/or to a "login" web page and/or if the user gives explicit permission. UBIs, depending on their type, might also be unique per website, as opposed to being the same regardless of the retrieving party.

6.3 UBI and Cookies

It could be argued that the functionality of a UBI could just as easily be implemented through the use of a special cookie. However, as we discuss below, there are a number of reasons why the UBI offers desirable features not accessible through the simple use of cookies. Of course the UBI would not replace cookies, as they serve as a means of adding state to HTTP, an otherwise stateless protocol.

One major difference between cookies and the proposed UBI is that the user has no means of controlling how cookies are used; all a user can do is have them deleted. By contrast, the UBI has a well-defined role and its use could be configured to meet user privacy requirements. That is, by providing explicit browser support for the UBI, its use can be controlled much more precisely than would be the case for cookies.

To help enable transparency of use and make websites accountable for their actions, browsers could usefully maintain a log of accesses to each type of UBI, e.g. including access date/time and accessing URL, just as is the case for cookies. However, unlike the case for cookies, it would also be helpful to log details of whether the requesting site is a third-party site, while also identifying the first-party site whose website contained the third-party link. Recording this additional information is made possible by implementing the UBI as a distinct browser feature.

Browsers should also enforce the use of HTTPS for UBI transfers, i.e. preventing access via HTTP. Currently, unless the appropriate flag is set, cookies can be transferred via HTTP, and transferring the UBI unencrypted would clearly be a privacy risk. It is interesting to observe that browser fingerprinting information is currently often transferred using HTTP (see 5.6).

Currently the expiry date of cookies is controlled by the server that created them, whereas UBI expiry would ideally be user-controllable. Moreover, if a multi-UBI system was implemented, UBIs would serve a range of purposes, and their behaviour and handling could be managed individually.

6.4 Privacy Considerations

Ideally, the use and functioning of a UBI should be standardized by an official body rather than being left to an initiative by a browser vendor. As mentioned in 6.1, Google considered replacing the use of cookies with a browser generated ID. Had such functionality been unilaterally added, perhaps with minimal user controls, it would potentially have given Google even greater control over online personalized advertising, to the detriment of user privacy, given that Google currently [8, 24] also owns the largest shares of both online advertising and browser users.

It might be argued that potential sharing of a UBI amongst trackers might increase privacy concerns. Analogously, concerns about sharing of browser fingerprinting IDs have recently been expressed [10]. However, in the case of UBIs, this might be avoided if the recommendations below and those in 6.1 and 6.3 are followed.

To help minimize the threat posed to user privacy, we propose that the following rules governing UBIs should be enforced by the browser.

- **UBI Access Control**: users should have full control over the use of all UBIs. This can be through prompts as well as enabling blacklisting/whitelisting of websites. Usage control should include providing means to enable/disable UBI access to third party websites. Denied websites could be provided with a dummy ID to prevent them learning that the user has denied them access.

- **UBI Reset**: like the existing advertising IDs, UBIs should be resettable, allowing users to opt to be forgotten by all websites that possess any of the previous UBI(s).

- **UBI Opacity**: UBIs should be generated in a way that reveals no information about its platform/client, nor linking with previously generated values.

6.5 Possible Issues

One possible criticism of the proposed UBI is that it could act as a sort of supercookie posing a significant threat to user privacy. The dangers can be seen by considering real-world use of advertising IDs (see 6.1). Google has stated that an "advertising identifier must not be connected to personally-identifiable information or associated with any persistent device identifier (for example: SSAID, MAC address, IMEI, etc.) without explicit consent of the user"[22]. However, recent research suggests that there is significant abuse of IDs of various types to create a very persistent ID that defeats

[22] `https://play.google.com/about/monetization-ads/ads/ad-id/` [accessed 12/10/2019]

the purpose of the advertising ID[23]. This suggests that third parties cannot be trusted to obey limitations on ID use unless restrictions are enforced by the browser itself. There are also potential issues with "popup fatigue", if a user is continually being asked about the use of IDs. These issues will certainly need to be properly addressed before browser support for UBIs is introduced.

Moreover, as noted in 5.7, browser fingerprinting plays an important role in the difficult balance between user privacy and the need for websites to sell advertising to pay for otherwise "free" services. Advertisers wish to track users to enable them to provide better-targeted advertising, whilst at the same time users want greater privacy protection. Browser vendors have the power to limit fingerprinting, but the economic justification for dedicating resources to fix the privacy problems is lacking. Of course, the whole idea of the UBI is to help achieve a better balance; other possible attempts to improve the situation include solutions such as the *basic attention token* (BAT)[24]. However, BAT aims to harmonize a currently dysfunctional advertising ecosystem[24], while the main objective of UBI is to enable users to assert some control over online tracking, and that regardless of whether or not it is being used for advertising purposes.

7 Conclusions

7.1 Browsers with Fingerprinting-Resisting Features

Some initial steps have been taken by certain browser vendors to tackle browser fingerprinting. However, these steps remain small and many browser vendors have not added any such features to their browsers. Firefox calls its techniques of this type *resistFingerprinting*. Using the term "resist" seems appropriate given that the measures certainly do not prevent it altogether. Firefox version 62 incorporates four options that can be enabled to resist fingerprinting, although some are disabled by default.

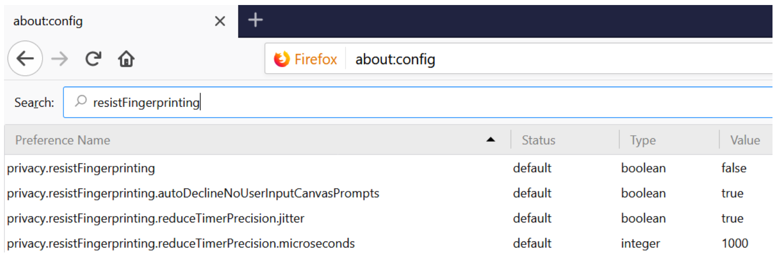
It seems that Firefox's fingerprinting features are not all listed or fully explained on its website[25]. One of these features prompts users if it detects potential canvas fingerprinting and another apparently reduces time-reporting precision. It is stated on Firefox's official website that the fingerprinting protection is still experimental and may negatively affect the browsing experience. Therefore, Firefox has opted not to make these options available in the main *Options* menu; instead, access is via the somewhat obscure advanced options, that are only accessible by typing `about:config` in the

[23]`https://blog.appcensus.io/2019/02/14/ad-ids-behaving-badly/` [accessed 12/10/2019]

[24]`https://basicattentiontoken.org` [accessed 18/10/2019]

[25]`https://support.mozilla.org/en/kb/firefox-protection-against-fingerprinting` [accessed 12/10/2019]

address bar (see Figure 1).



Preference Name	Status	Type	Value
privacy.resistFingerprinting	default	boolean	false
privacy.resistFingerprinting.autoDeclineNoUserInputCanvasPrompts	default	boolean	true
privacy.resistFingerprinting.reduceTimerPrecision.jitter	default	boolean	true
privacy.resistFingerprinting.reduceTimerPrecision.microseconds	default	integer	1000

Figure 1: Firefox anti-fingerprinting options

Apart from these limited measures in Firefox, only Safari and the Tor browser appear to contain significant anti-fingerprinting features. Even in this case, only the latest Safari version (i.e. Safari 12) possesses such features. This leaves around more than 90%[26] of web users without any default anti-fingerprinting protection. The Safari and Tor browser implementations of anti-fingerprinting measures differ, although both Safari and Tor are designed to make all versions indistinguishable.

7.2 A Possible Role for Regulation

As discussed in 5, a number of steps could be taken by browser vendors to make fingerprinting significantly less effective, including removing (or limiting) uncommonly used APIs and reducing API accuracy. The main question is how to persuade browser vendors to implement such steps. One possible route might involve some kind of regulation.

More than 92%[26] of users use one of the five most widely used browsers, but trackers and the domains they use are very numerous. It therefore seems reasonable to focus more on controlling browser fingerprinting by regulating browsers rather than by regulating websites. Moreover, browsers act as a kind of middle man between websites and users, and can thus act as a type of privacy regulator [23]. However, it is important to note that some browser vendors also make extensive use of browser fingerprinting, and are thus likely to be reluctant to restrict it. Finally, despite the primary focus on browsers, the regulation of websites remains an important possibility, especially given that it appears that most tracking is performed by a limited number of third-parties [3].

7.3 Final Remarks

It is unlikely that browser fingerprinting (or variants of it) will disappear any time soon. Studies have shown that its deployment is increasing. Moreover,

[26]`https://netmarketshare.com/browser-market-share.aspx` [accessed 15/10/2019]

the methods used to perform it are continually changing as browsers themselves evolve, making controlling it very difficult. Also, fingerprinting does have uses other than for tracking and so eradicating it without providing an alternative (e.g. as provided by UBIs) is likely to be infeasible in practice. In summary, it seems that unless effective alternatives are provided, preventing browser fingerprinting is likely to be impossible. Given the threat it poses to user privacy, this is clearly an important topic for future research.

References

[1] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juárez, Arvind Narayanan, and Claudia Díaz. The web never forgets: Persistent tracking mechanisms in the wild. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3–7, 2014*, pages 674–689. ACM, 2014.

[2] Nasser Mohammed Al-Fannah and Wanpeng Li. Not all browsers are created equal: Comparing web browser fingerprintability. In Satoshi Obana and Koji Chida, editors, *Advances in Information and Computer Security — 12th International Workshop on Security, IWSEC 2017, Hiroshima, Japan, August 30 – September 1, 2017, Proceedings*, volume 10418 of *Lecture Notes in Computer Science*, pages 105–120. Springer, 2017.

[3] Nasser Mohammed Al-Fannah, Wanpeng Li, and Chris J. Mitchell. Beyond cookie monster amnesia: Real world persistent online tracking. In Liqun Chen, Mark Manulis, and Steve Schneider, editors, *Information Security — 21st International Conference, ISC 2018, Guildford, UK, September 9–12, 2018, Proceedings*, volume 11060 of *Lecture Notes in Computer Science*, pages 481–501. Springer, 2018.

[4] Károly Boda, Ádám Máté Földes, Gábor György Gulyás, and Sándor Imre. User tracking on the web via cross-browser fingerprinting. In Peeter Laud, editor, *Information Security Technology for Applications — 16th Nordic Conference on Secure IT Systems, NordSec 2011, Tallinn, Estonia, October 26-28, 2011, Revised Selected Papers*, volume 7161 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2011.

[5] Rik Cabanier, Ian Hickson, Tom Wiltzius, Jatinder Mann, and Jay Munro. HTML canvas 2D context. W3C recommendation, W3C, November 2015. `http://www.w3.org/TR/2015/REC-2dcontext-20151119/`.

[6] Nick Doty. Fingerprinting guidance for web specification authors (draft). W3C note, W3C, November 2015. `https://www.w3.org/TR/fingerprinting-guidance/`.

[7] Peter Eckersley. How unique is your web browser? In *Privacy Enhancing Technologies, 10th International Symposium, PETS 2010, Berlin, Germany, July 21-23, 2010. Proceedings*, pages 1–18, 2010.

[8] eMarketer Inc. Global ad spending update. `https://www.emarketer.com/content/global-ad-spending-update`, 2019. Online; accessed 25/02/2019.

[9] Amin FaizKhademi, Mohammad Zulkernine, and Komminist Weldemariam. FPGuard: Detection and prevention of browser fingerprinting. In Pierangela Samarati, editor, *29th Annual IFIP WG 11.3 Working Conference, DBSec 2015, Fairfax, VA, USA, July 13–15, 2015*, volume 9149 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2015.

[10] Marjan Falahrastegar, Hamed Haddadi, Steve Uhlig, and Richard Mortier. Tracking personal identifiers across the web. In Thomas Karagiannis and Xenofontas A. Dimitropoulos, editors, *Passive and Active Measurement - 17th International Conference, PAM 2016, Heraklion, Greece, March 31 – April 1, 2016. Proceedings*, volume 9631 of *Lecture Notes in Computer Science*, pages 30–41. Springer, 2016.

[11] I. Fette and A. Melnikov. Privacy considerations for internet protocols. RFC 6973, RFC Editor, July 2013. `https://tools.ietf.org/rfc/rfc6973.txt`.

[12] Roy T. Fielding and Julian F. Reschke. Hypertext transfer protocol (HTTP/1.1): semantics and content. RFC 7231, RFC Editor, June 2014. `https://tools.ietf.org/rfc/rfc7231.txt`.

[13] Ugo Fiore, Aniello Castiglione, Alfredo De Santis, and Francesco Palmieri. Countering browser fingerprinting techniques: Constructing a fake profile with google chrome. In *17th International Conference on Network-Based Information Systems, NBiS 2014, Salerno, Italy, September 10–12, 2014*, pages 355–360. IEEE Computer Society, 2014.

[14] Alejandro Gómez-Boix, Pierre Laperdrix, and Benoit Baudry. Hiding in the crowd: an analysis of the effectiveness of browser fingerprinting at large scale. In Pierre-Antoine Champin, Fabien L. Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis, editors, *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23–27, 2018*, pages 309–318. ACM, 2018.

[15] Business Insider. Google and facebook to replace cookies, 2014. `https://www.businessinsider.com/google-and-facebook-to-replace-cookies-2014-2?r=US&IR=T` accessed 19/12/2018.

[16] Sheharbano Khattak, David Fifield, Sadia Afroz, Mobin Javed, Srikanth Sundaresan, Damon McCoy, Vern Paxson, and Steven J. Murdoch. Do you see what I see? differential treatment of anonymous users. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016.

[17] Mounir Lamouri and Anssi Kostiainen. Battery status API. Candidate recommendation, W3C, July 2016. `http://www.w3.org/TR/2016/CR-battery-status-20160707/`.

[18] Pierre Laperdrix, Benoit Baudry, and Vikas Mishra. FPRandom: Randomizing core browser objects to break advanced device fingerprinting techniques. In Eric Bodden, Mathias Payer, and Elias Athanasopoulos, editors, *Engineering Secure Software and Systems — 9th International Symposium, ESSoS 2017, Bonn, Germany, July 3-5, 2017, Proceedings*, volume 10379 of *Lecture Notes in Computer Science*, pages 97–114. Springer, 2017.

[19] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. Mitigating browser fingerprint tracking: Multi-level reconfiguration and diversification. In *10th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2015, Florence, Italy, May 18–19, 2015*, pages 98–108, 2015.

[20] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. In *IEEE Symposium on Security and Privacy, S&P 2016, San Jose, CA, USA, May 22–26, 2016*, pages 878–894. IEEE Computer Society, 2016.

[21] S. Luangmaneerote, E. Zaluska, and L. Carr. Survey of existing fingerprint countermeasures. In *2016 International Conference on Information Society (i-Society)*, pages 137–141. IEEE Computer Society, October 2016.

[22] Keaton Mowery and Hovav Shacham. Pixel perfect: Fingerprinting canvas in HTML5. In Matt Fredrikson, editor, *W2SP 2012, San Francisco, CA, USA*. IEEE Computer Society, May 2012. `http://www.ieee-security.org/TC/W2SP/2012/papers/w2sp12-final4.pdf`.

[23] Arvind Narayanan and Dillon Reisman. The princeton web transparency and accountability project. In Tania Cerquitelli, Daniele Quercia, and Frank Pasquale, editors, *Transparent Data Mining for Big and Small Data*, pages 45–67. Springer International Publishing, 2017.

[24] NetMarketShare. Browser market share. `https://netmarketshare.com/browser-market-share.aspx`, 2019. Online; accessed 25/02/2019.

[25] Nick Nikiforakis, Wouter Joosen, and Benjamin Livshits. Privaricator: Deceiving fingerprinters with little white lies. In Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi, editors, *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18–22, 2015*, pages 820–830. ACM, 2015.

[26] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19–22, 2013*, pages 541–555. IEEE Computer Society, 2013.

[27] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. On the workings and current practices of web-based device fingerprinting. *IEEE Security & Privacy*, 12(3):28–36, June 2014.

[28] Lukasz Olejnik, Gunes Acar, Claude Castelluccia, and Claudia Díaz. The leaking battery — A privacy analysis of the HTML5 battery status API. In *Data Privacy Management, and Security Assurance — 10th International Workshop, DPM 2015, and 4th International Workshop, QASA 2015, Vienna, Austria, September 21–22, 2015. Revised Selected Papers*, volume 9481 of *Lecture Notes in Computer Science*, pages 254–263. Springer, 2015.

[29] Lukasz Olejnik, Steven Englehardt, and Arvind Narayanan. Battery status not included: Assessing privacy in web standards. In *3rd International Workshop on Privacy Engineering (IWPE'17), San Jose, USA*, March 2017.

[30] Mike Perry. Do not beg: Moving beyond DNT through privacy by design. W3C DNT, Tor Project, November 2012. `https://www.w3.org/2012/dnt-ws/position-papers/21.pdf`.

[31] Mike Perry, Erinn Clark, and Steven Murdoch. The design and implementation of the tor browser. Draft, The Tor Project, June 2013. `https://www.torproject.org/projects/torbrowser/design/`.

[32] Lindsay Simpkins, Xiaohong Yuan, Jwalit Modi, Justin Zhan, and Li Yang. A course module on web tracking and privacy. In Michael E. Whitman and Humayun Zafar, editors, *Proceedings of the 2015 Information Security Curriculum Development Conference, InfoSecCD 2015, Kennesaw, GA, USA, October 10, 2015*, pages 10:1–10:7. ACM, 2015.

[33] David Singer and Roy Fielding. Tracking preference expression (DNT). WD not longer in development, W3C, January 2019. https://www.w3.org/TR/2019/NOTE-tracking-dnt-20190117/.

[34] Peter Snyder, Lara Ansari, Cynthia Taylor, and Chris Kanich. Browser feature usage on the modern web. In *Proceedings of the 2016 ACM on Internet Measurement Conference, IMC 2016, Santa Monica, CA, USA, November 14–16, 2016*, pages 97–110. ACM, 2016.

[35] Oleksii Starov and Nick Nikiforakis. XHOUND: quantifying the fingerprintability of browser extensions. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22–26, 2017*, pages 941–956. IEEE Computer Society, 2017.

[36] Christof Ferreira Torres, Hugo L. Jonker, and Sjouke Mauw. FP-Block: Usable web privacy by controlling browser fingerprinting. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *Computer Security — ESORICS 2015 — 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21–25, 2015, Proceedings, Part II*, volume 9327 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2015.

[37] Antoine Vastel, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. Fp-scanner: The privacy implications of browser fingerprint inconsistencies. In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15–17, 2018.*, pages 135–150. USENIX Association, August 2018.

[38] Ting-Fang Yen, Yinglian Xie, Fang Yu, Roger Peng Yu, and Martín Abadi. Host fingerprinting and tracking on the web: Privacy and security implications. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*, February 2012.