Imane Fouad, Cristiana Santos, Arnaud Legout, and Nataliia Bielova

My Cookie is a phoenix: detection, measurement, and lawfulness of cookie respawning with browser fingerprinting

Abstract: Stateful and stateless web tracking gathered much attention in the last decade, however they were always measured separately. To the best of our knowledge, our study is the first to detect and measure cookie respawning with browser and machine fingerprinting. We develop a detection methodology that allows us to detect cookies dependency on browser and machine features. Our results show that 1,150 out of the top 30,000 Alexa websites deploy this tracking mechanism. We find out that this technique can be used to track users across websites even when third-party cookies are deprecated. Together with a legal scholar, we conclude that cookie respawning with browser fingerprinting lacks legal interpretation under the GDPR and the ePrivacy directive, but its use in practice may breach them, thus subjecting it to fines up to 20 million €.

Keywords: fingerprinting, cookie respawning, GDPR

DOI 10.56553/popets-2022-0063
Received 2021-11-30; revised 2022-03-15; accepted 2022-03-16.

1 Introduction

In the last decades, the usage of the web has considerably increased, along with the web browsers sophistication. In parallel, numerous companies built their business models on tracking web users. Therefore, browsers evolution does not only provide a better user experience, but also allows the emergence of new tracking techniques exploited by companies to collect users' data.

In the literature, two main categories of tracking techniques have been studied: stateful and stateless. *Stateful tracking* is a standard technique that relies on browser storage such as cookies [2, 8, 25, 55]. Trackers store a unique identifier in the cookie and later use it to recognize a user and track their activity across, possibly, different websites. The simplest way to protect from such tracking is to delete the unique identifier by, e.g., cleaning the cookie storage. However, trackers can recreate deleted cookies using a technique called *cookie respawning* to track users. For instance, a tracker can use multiple browser storages that store identifiers, in addition to the cookie storage, such as the HTML5 localStorage [8]. Consequently, even if the user cleans the cookie storage, the tracker can still recreate cookies using other storages [2, 8, 55, 63].

Stateless tracking allows for tracking users without storing identifiers in their browser storage. Using *browser fingerprinting* [3, 14, 24, 33, 36, 50], trackers can identify a user through a combination of the user's browser and machine features, such as the user agent or IP address. Whereas it is hard to prevent it, browser fingerprinting is not stable over time. Vastel et al. [70] showed that fingerprints change frequently: out of 1,905 studied browser instances, 50% changed their fingerprints in less than 5 days, and 80% in less than 10 days. This instability is caused either by automatic triggers such as software updates or by changes in the user's context such as travelling to a different timezone.

In summary, stateful tracking is a stable way to track users until they clean cookies and other browser storages. Stateless tracking is not stable over time, but does not require any storage and can not be easily stopped by the user. So given that each technique is not perfect, *how can a tracker take advantage of the best of the two worlds?* The tracker can first use a browser fingerprint to create an identifier and store it in the browser's cookie. In this way, even if a user cleans this cookie, the identifier can be recreated with a browser fingerprint. Moreover, even if the fingerprint changes

Imane Fouad: Univ. Lille, CNRS, Inria, E-mail: imane.fouad@inria.fr
Cristiana Santos: Utrecht University, E-mail: cristianasantos@protonmail.com
Arnaud Legout: Inria, E-mail: arnaud.legout@inria.fr
Nataliia Bielova: LINC team, CNIL, France, E-mail: nataliia.bielova@inria.fr. The work conducted by Nataliia Bielova was accomplished while she was at Inria PRIVATICS team prior to her joining the CNIL on 1 September 2021. The views expressed in this paper does not necessarily reflect the view of the commission or any individual commissioner.

over time, the identifier stored in the cookie can help to match the new fingerprint with the old fingerprint of the same user. We refer to this tracking technique as *cookie respawning with browser fingerprinting*.

Several studies measured the prevalence of stateful [2, 25, 55] or stateless [3, 14, 24, 50] tracking techniques separately. However, to the best of our knowledge, *we are the first to study how trackers profit from combining both stateful and stateless techniques*.

The aim of this paper is to propose a robust methodology to detect and measure the prevalence of cookie respawning with browser fingerprinting, followed by a technical and legal analysis of the privacy implications of this tracking technique. In this paper, we make the following contributions.

1) We designed a robust method to identify which features are used to respawn a cookie. Our contribution lies in the design of a method to automatically identify the set of fingerprinting features used to generate a cookie, hence, to conclude which user information is collected. We additionally perform a permutation test (N=10,000, $p<0.05$) to provide certainty on the dependency between the features and the cookies.

2) We conduct the first study of cookie respawning with browser fingerprinting. We show that the stateful and stateless tracking techniques that were studied separately are, in fact, actively used together by trackers. We found that $1,150$ (3.83%) of the Alexa top $30,000$ websites use cookie respawning with browser fingerprinting.

3) We show that cookie respawning with browser fingerprinting is highly deployed in popular websites. Cookie respawning with browser fingerprinting is also happening on websites from different categories including highly sensitive ones such as adult websites.

4) We show that cookie respawning with browser fingerprinting lacks legal interpretation and its use, in practice, violates the GDPR and the ePrivacy directive. We are the first to assess the legal consequences of this practice together with a legal expert co-author. Despite the intrusiveness of this practice, it has been overlooked in the EU Data Protection Law and it is not researched in legal scholarship, nor audited by supervisory authorities.

In this paper, we use the term *tracker* to refer to any company trying to track user's behavior. This tracking behavior can range from analytics to cross site tracking, with very different privacy consequences for the end-users. Whereas this paper does not focus of cross-site tracking, the *cookie respawning with browser fingerprinting* technique can be used for any kind of tracking activities, but more importantly, it has the potential to evade user protection techniques against cross-site tracking such as the yet to be deployed third party cookie deprecation [58] (see Section 5.5).

2 Background

2.1 Scope of cookies: host and owner

In this paper, we make a distinction between the notion of cookie *host* and cookie *owner*. When a cookie is stored in the browser, it is identified by a tuple (*host*, key, value). If the cookie is set via an HTTP(S) response header, then the *host* of the cookie represents a domain that sets the cookie. However, when the cookie is set programmatically via a JavaScript script included in the website, the script gets executed in the context, or "origin" where it is included. Due to the Same Origin Policy (SOP) [57], the *host* of a cookie set by the script is the origin of the execution context of the script, and not the domain that contains the script. Given a cookie stored in the browser with its (*host*, key, value), when a browser sends a request to a domain, it attaches a cookie to the request if the cookie *host* matches the domain or the subdomain of the request [44].

A cookie *owner* is responsible for setting the cookie. It is either a domain that sets a cookie via HTTP(S) response header (and in this case, matches with the cookie *host*), or the domain that hosts a script that sets the cookie programmatically (generally speaking, here the owner is different from host). For example, `site.com` includes a third party script from `tracker.com`. After loading, the script sets a cookie in the context of the visited website `site.com`. In this case, the cookie owner is `tracker.com`, but the cookie host is `site.com`.

2.2 Web tracking technologies

Cookie-based tracking. Websites are composed of first party content and numerous third-party content, such as advertisements, web analytic scripts, social widgets, or images. Following the standard naming [40], for a given website we distinguish two kinds of domains: *first-party* domain that is the domain of the website, and *third-party* domains that are domains of the *third-party* content served on the website.

Using HTTP(S) request (or response), any content of the webpage can set (or receive) cookies. Addition-

ally, cookies can be set programmatically via a script. Every cookie is stored in the browser with a domain referred to as host, and path, so that every new HTTP(S) request sent to the same domain and path gets a cookie associated thereto attached to the request. First-party cookies set by first-party domains are capable to track users *within the same website*. Third party cookies set by third-party domains allow third parties to track users *cross-websites* [55].

Browser fingerprinting. Browser fingerprinting is a stateless tracking technique that provides the ability to identify and track users without using their browser storage [3, 5, 14, 22], unlike cookie-based tracking. When a user visits a web page that includes a fingerprinting script, this script will return to a fingerprinter server a list of features composed of user's browser and machine characteristics, such as user agent or time zone. The trackers use these collected features to build a unique identifier.

Cookie respawning. Cookie respawning is the process of automatically recreating a cookie deleted by the user (usually by cleaning the cookie storage). Several techniques can be used to respawn a cookie. While related works focused on exploiting another browser storage (e.g., the HTML5 local storage) that duplicates the information contained in the cookie [3, 8, 63], in this work, we focus on the usage of a browser fingerprint to recreate a cookie. Section 4 describes how a tracker can exploit a browser fingerprint to respawn a cookie.

3 Related work

Cookie based tracking is a classical tracking technique which has been widely studied [3, 8, 22, 24, 40, 50, 51, 54, 55, 63], and is now commonly blocked by modern browsers [34, 49] and add-ons [19, 31]. In this paper, we explore a more sophisticated technique combining cookie based tracking with fingerprinting.

In 2010, the Panopticlick study showed that fingerprints can be potentially used for web tracking [22]. Following this study, several fingerprinting tracking techniques were discovered, and showed that user's browser and machine features can be deployed to track their activity [3, 5, 14, 62].

The term *respawning* was first introduced in 2009 by Soltani et al. [63]. They showed that trackers are abusing the usage of the Flash cookies in order to respawn or recreate the removed HTTP cookies. This work attracted general audience attention [45, 46] and triggered lawsuits [38, 39]. Following Soltani et al. work, other studies started analyzing the usage of other storages for respawning such as ETags and localStorage [8]. Sorensen studied the usage of browser cache in cookie respawning [65]. Acar et al. automated the detection of cookie respawning and found that IndexedDB (a client-side storage) can be used to respawn cookies as well [3]. Roesner et al. showed that cookies can be respawned from local and Flash storages [55].

Laperdrix et al. [36] surveyed recent advancement in measurement and detection of browser fingerprinting. The survey mentions [36, §5.1] that browser fingerprint together with IP address can be used to regenerate deleted cookies, however no previous work studied this phenomena.

Unlike previous works that studied the usage of browser storages to respawn cookies, or measured fingerprinting independently, our study analyzes the usage of fingerprinting to respawn cookies.

4 Methodology

When a user visits a web page with some content located on a tracker's server, the user's browser sends an HTTP(s) request to the server to fetch this content. This request contains several HTTP headers, such as user agent, and an IP address that tracker's server receives *passively*. We refer to such information as *passive features*. To collect additional information, the tracker can include in the visited web page a script that gets executed on the user's browser. The script retrieves multiple browser and machine information, such as the time zone, and sends them to a server of the remote tracker. We refer to such information as *active features*. In the following, we define a *browser fingerprint* as the set of active and passive features accessed by the tracker.

We say that a tracker *respawns a cookie* when it recreates the exact same cookie after the user revisits the website in a clean browser.

4.1 How to benefit from a combination of cookies and fingerprinting?

To benefit from both techniques, the tracker can first use a browser fingerprint to create an identifier and store it in the browser's cookie. In this way, even if a user cleans this cookie, the identifier can be recreated with a browser fingerprint. Moreover, even if the fingerprint

changes over time, the identifier stored in the cookie can help to match the new fingerprint with the old fingerprint of the same user. We explain these scenarios and benefits in details below.

Figure 1(a) shows that the tracker first receives a set of user's active and\or passive features (step ①). In step ②, the tracker generates an identifier from the received features, that it might store on the server's matching table. The tracker then stores the created identifier in the user's browser cookie, either via the `Set-cookie` header (step ③) or programmatically via JavaScript (not shown in Figure 1(a)). As a result, an identifier is stored in the browser's cookie database (step ④).

Figures 1(b) shows what happens when the user does not have a cookie *123* in the browser, however the fingerprint *fp456* remains the same. In this case, the fingerprint *fp456* is sent to the server of `tracker.com` (step ⑤), and it allows the tracker to match the known fingerprint and the cookie previously set for this user (step ⑥). As a result, the tracker is able to set again the same cookie *123*, previously deleted by the user (step ⑦). This allows the tracker to respawn deleted user cookies with browser fingerprinting and continue tracking the user via such cookies (step ⑧).

Figure 1(c) presents the consequences of cookie respawning with browser fingerprinting. When the browser fingerprint of the user is updated from *fp456* to *fp789*, the server of `tracker.com` receives an old cookie *123* with a new fingerprint *fp789* (step ⑨). The cookie *123* helps the server to recognize the user's browser and update the corresponding record in the matching table and substitute a fingerprint *fp456* to *fp789* associated to cookie *123* (step ⑩). This allows the tracker to match different fingerprints of the same user, given that fingerprinting is not stable over time.

As a result, cookie respawning with browser fingerprinting allows trackers to respawn deleted cookies, and also to link different browser fingerprints of the same user. This makes the tracking robust to either cookie deletion or fingerprint change. Only in case the browser fingerprint changes and the cookie is deleted at the same time, the tracker will not be able to recognize the user and hence to continue tracking this user.

In this paper, we propose a robust methodology to detect the mechanisms presented in Figures 1(a) and 1(b). In this section, we first introduce our methodology to crawl Alexa top 30,000 websites (Section 4.2). Next, we present our method to detect cookie respawning with browser fingerprinting (Section 4.3). Then, we describe the fingerprinting features used in our study and spoofing techniques (Section 4.4). Finally, we list the limitations of our methodology (Section 4.5).

4.2 Measurement setup

We performed passive web measurement on March 2021 of the Alexa top 30,000 websites extracted on March 2020[1]. All measurements are performed using the OpenWPM platform [52] on the Firefox browser. We used two machines to perform the crawls in our study. The versions of OpenWPM and Firefox, the time period of the crawl, and the characteristics of the two machines used in this study are presented in Table 7 of the Appendix A.

We used different characteristics with two machines so that they appear as different users, as done by previous works [2, 24, 25, 30]. Ideally, we would have used two distinct machines with different locations to detect user specific cookies, however, both machine A and machine B are located in France. Hence, to change the Machine B geolocation, we spoofed the parameters latitude and longitude by modifying the value of `geo.wifi.uri` advanced preference in the browser and point it to Alaska.

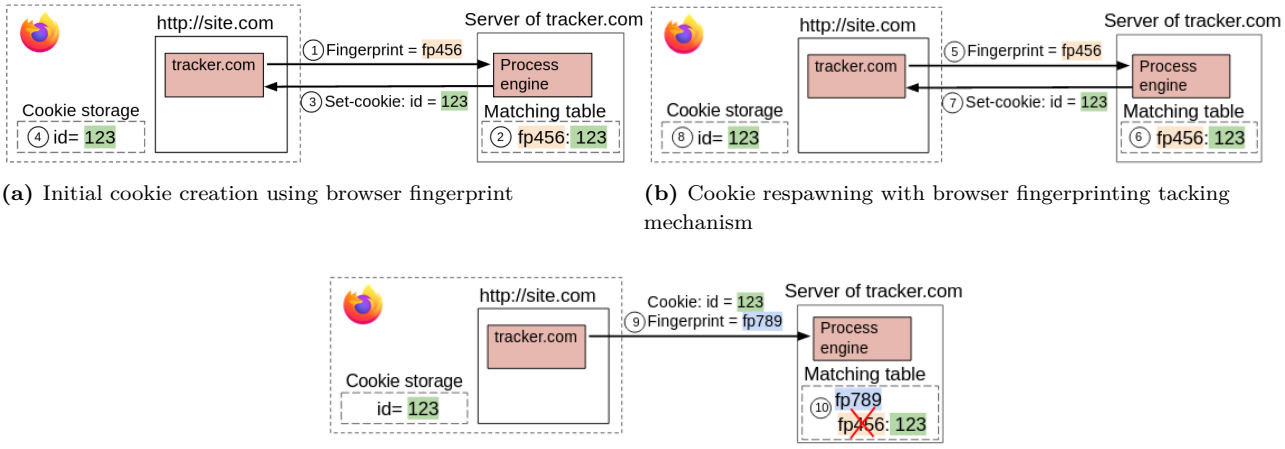
All our crawls are based on the notion of *stateless crawling instances*. We define a stateless crawling instance of a website X as follows: (1) we visit the home page of the website X and keep the page open until all content is loaded to capture all cookies stored (we set the timeout for loading the page to 90s), (2) we clear the profile by removing the Firefox profile directory that includes all cookies and browser storages. The rationale behind the stateless crawling instance is to ensure that we do not keep any state in the browser between two crawling instances. This guarantees that respawned cookies do not get restored from other browser storages.

We perform stateless crawling instances of the Alexa top 30,000 websites and for *each stateless crawling instance*, we extracted HTTP and scripts information automatically collected by OpenWPM (see list of recorded data in Appendix B).

4.3 Detecting cookie respawning with browser fingerprinting with sequential crawling

Figure 2 presents our sequential crawling methodology that detects which fingerprinting features are used to

1 We made this list of websites publicly available [6].



(a) Initial cookie creation using browser fingerprint

(b) Cookie respawning with browser fingerprinting tacking mechanism

(c) Usage of cookies to ensure fingerprint stability

Fig. 1. Cookie respawning with browser fingerprinting tracking technique. (a) (step 1) The tracker receives user's features, (step 2) then stores a fingerprint *fp456* associated with the features and generates a corresponding cookie *123*. (step 3) Next, the tracker sets the cookie in the user's browser. (step 4) As a result an identifier is stored in the browser cookie storage. (b) When the user cleans the browser and revisit the website, (step 5) the tracker receives the fingerprint *fp456*, (step 6) extracts the corresponding cookie from the matching table, (step 7) and re-sets it in the user's browser. (step 8) As a result, the cookie *123* is recreated in the user's browser. (c) The fingerprint is not stable over time, (step 9) thus the user fingerprint might change. (step 10) The tracker can use the cookie received with the fingerprint to update the latest on the server side.

respawn cookies. Our method consists of two main steps explained in this section:

- **Create the initial set of candidate respawned cookies:** we identify candidate respawned cookies by collecting all cookies that get respawned in a clean browsing instance, and we remove cookies that are not user-specific.
- **Identify dependency of each respawned cookie on each fingerprinting feature:** we spoof each feature independently to detect whether the value of a respawned cookie has changed when the feature is spoofed. We perform a permutation test ($N = 10,000$, $p < 0.05$) to add statistical evidence on the dependency between a feature and the respawned cookie.

4.3.1 Creation of the initial set of candidate respawned cookies

To build the initial set of candidate respawned cookies, we perform two stateless crawling instances from machine A as described in Figure 2 (*Initial* and *Reappearance crawl*). Via these two crawls, we ensure that all browser storages are cleaned and the only way for cookies to be respawned is with browser fingerprinting.

We define a cookie as the tuple *(host, key, value)* where *host* is the domain that can access the cookie. To create the set of candidate respawned cookies, we only collect cookies that appear in both the *Initial* and *Reappearance crawl* when visiting the same website in the two crawls. Note that due to our sequential crawling (we visit websites in a sequence), we only consider candidate respawned cookies within the same website.

Previous research [2, 24, 25, 30] considered that cookies are non specific to the users and hence unlikely to be used for tracking when their values are identical for several users. Therefore, using distinct machines to remove non user-specific cookies became a common method in this research area. We follow this methodology and remove cookies that are not user-specific from our set of candidate respawned cookies. To do so, we performed an additional *User specific crawl* from a different machine B that appears to trackers as a different user. Practically, we perform the *Initial crawl* and *User specific crawl* in parallel, and the *Reappearance crawl* right after the *Initial crawl* completes (Figure 2). It is important that machines A and B have different fingerprinting features (see Table 7 of the Appendix) to avoid wrong categorization of cookies that depend on these features as non user-specific.

We hence remove the following cookies from the set of candidate respawned cookies and keep only user-specific cookies:
1) a cookie *(host, key, value)* if it appears on both the *Initial crawl* on machine A and *User specific crawl* on machine B with the same *host*, *key*, and *value*.
2) a cookie *(host, key, value)* if a cookie with the same *host* and *key* is not present in a *User specific crawl*. We adopt a conservative strategy to remove such cookies because we do not have a proof that such cookies are user-specific.

We remove cookies that are not user-specific or do not re-appear to ensure that only user-specific cookies are further analysed.

4.3.2 Identifying dependency of each respawned cookie on each fingerprinting feature

The set of candidate respawned cookies contains cookies that are both user-specific and respawn when crawled a second time after we used a new browser instance with a cleaned browser storage. Therefore, cookies in this set are very likely to be respawned with the use of browser fingerprinting. To detect whether fingerprinting features are used to respawn the collected cookies, we performed the following steps. We first identified 8 fingerprinting features from previous research (see more details on the choice of features and methods to spoof them in Section 4.4). Then, for each website u where we have at least one candidate respawned cookie, we perform 99 crawls: 11 spoofing crawls per studied fingerprinting feature f, and 11 crawls with all features set to their initial values (as in *Initial crawl*) that we refer to as *Control crawls*. In each of the total 88 spoofing crawls, we first spoof the feature f and perform a stateless *Spoofing crawl* of the website u. For each user-specific respawned cookie from the candidate set, we perform the following algorithm.

- For each of the 88 crawls, we label the cookie as respawned if the cookie's *host* and *key* are identical but *value* are different from the Initial crawl. As a result we get 11 observations for each feature.
- For every feature, we perform a permutation test with the 11 observations from the *Control crawls* using 10,000 permutations. The statistical test assess the difference of the probability to have the cookie respawned between the *Spoofing crawls* and the *Control crawls*.
- We consider that the cookie is *feature dependent* if the p-value for the test statistic is lower than 0.05.

4.3.3 Summary of the crawling methodology

We consider a cookie to be user specific if it appears in the initial crawl, reappears with the *same value* in the reappearance crawl performed from a clean browser instance, but does not reappear in the user-specific crawl that is performed from a different machine (see Table 7) than the two other crawls. These user specific cookies represent our set of candidate respawned cookies.

Then, we performed additional crawls to test the dependency of a cookie to a specific feature f. Ideally, a cookie respawned due to a feature f must reappear with the *same value* for each crawl when the feature f is not modified, and must not reappear when f is spoofed. However, due to the dynamic nature of the Web, the cookie might change independently of a feature, because the content of the page changed. In practice, we observed candidate respawn cookies that reappear for some but not all crawls in which a feature f is not modified. To show whether this reappearance is due to chance or due to a real dependency to the feature f, we performed a permutation test to show the statistical significance of the dependency to a feature f.

4.4 Selection of fingerprinting features and spoofing techniques

To achieve a high uniqueness of an identifier built from a browser fingerprint, trackers use a combination of both passive and active browser and machine features. Though browser features are useful for fingerprinting, using them alone might be problematic for trackers because of the usage of multiple browsers among users [14, 60, 69]. To improve the accuracy of the fingerprint, trackers also use machine related features such as the IP address, or the OS version [5, 11].

Table 1 presents a full list of studied browser and machine features that we selected based on the most common features in prior works on browser fingerprinting [2, 5, 11, 14, 18, 24, 32, 37, 48].

We used two methods to spoof fingerprinting features: Firefox preferences and add-ons. We validated that each feature is properly spoofed on our own testing website with a fingerprinting script and also by using *whoer* website [72] that verifies the information sent by the user's browser and machine to the web.

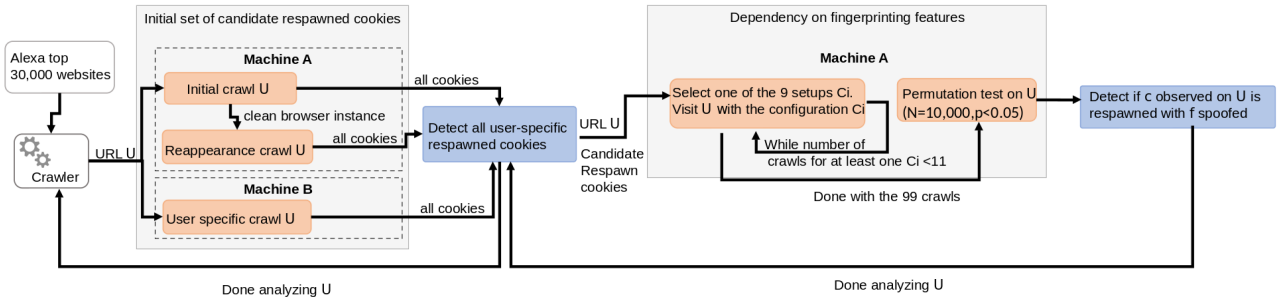


Fig. 2. Sequential crawling of $30,000$ top Alexa websites to identify cookie respawning with browser fingerprinting. For each website, we perform an *Initial crawl* from machine A and, a *User specific crawl* from machine B to detect machine unrelated cookies. After *Initial crawl* finishes, we start a *Reappearance crawl* from machine A to detect reappearance of cookies. Using *Initial crawl*, *User specific crawl*, and *Reappearance crawl* we detect *user-specific* cookies that reappear in *Reappearance crawl*, but not in *User specific crawl*. For such cookies, we randomly chose one configuration C_i: either spoof one feature at a time or to set all features to initial value. We perform 99 stateless crawls (11 *Spoofing crawl*s per feature and 11 *Control crawl*s where the studied features are unspoofed). Finally, we perform a permutation test for each feature (N=10,000), and we consider that the cookie is *feature dependent* if the resulting p-value < 0.05. All these steps are discussed in Section 4.3.

	Feature name	Feature type
Browser features	Accept language [5, 37]	Active/Passive
	Geolocation [5]	Active
	User agent [5, 11, 32, 37]	Active/Passive
	Do not track [32, 37]	Active/Passive
Machine features	WebGL [5, 14, 32, 37, 48]	Active
	Canvas [2, 14, 18, 24, 32, 37, 48]	Active
	IP address [5, 11, 18, 24]	Passive
	Time zone [11, 32, 37]	Active

Table 1. Studied fingerprinting features.

4.4.1 Spoofing using Firefox preferences

Firefox allows to change its settings in the browser preferences of *about:config* page. With this method, we spoofed the following features.

User agent. The User-Agent HTTP header allows the servers to identify the operating system and the browser used by the client. The *Initial crawl* runs in Firefox under Linux (see Table 7 in the Appendix for details). To spoof the user agent, we changed the `general.useragent.override` preference in the browser to Internet Explorer under Windows: *Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; AS; rv:11.0) like Gecko*. We checked the spoofing efficiency on our testing website with a fingerprinting script. The script returns the user agent value using the `navigator.userAgent` API. We tested the user agent value returned with the HTTP header using the *whoer* website [72]. We found that the user agent value was spoofed both in JavaScript calls and HTTP headers.

Geolocation. The geolocation is used to identify the user's physical location. The *Initial crawl* has as location *Cote d'Azur, France*. We spoofed the geolocation parameters latitude and longitude by modifying the value of `geo.wifi.uri` preference in the browser and point it to the *Time Square, US* ("lat": 40.7590, "lng": -73.9845). We validated the spoofing efficiency using a script call to `navigator.geolocation` API.

WebGL. The WebGL API is used to give information on the device GPU. In our study, we focus on the WebGL renderer attribute that precises the name of the model of the GPU. We spoofed the WebGL renderer using the `webgl.render-string-override` preference in the browser. We changed the value of WebGL renderer to *GeForce GTX 650 Ti/PCIe/SSE2*. To retrieve information about the graphic driver and read the WebGL renderer value, we used the *WEBGL_debug_renderer_info* add-on. We validated the WebGL spoofing efficiency by using the add-on in our customized website.

Do Not Track. The Do Not Track (DNT) header indicates user's tracking preference. Users can express that they don't want to get tracked by setting the DNT to *True*. In the *Initial crawl*, the DNT was set to *null*. We enabled the DNT header, and we set it to *True* using the `privacy.donottrackheader.enabled` preference. We validated that the DNT returned value in the HTTP header is set to True using the *whoer* website.

4.4.2 Spoofing using browser add-ons

The browser preferences do not provide a spoofing mechanism for all fingerprinting features. We used browser add-ons to spoof such features.

Canvas. The HTML canvas element is used to draw graphics on a web page. The difference in font rendering, smoothing, as well as other features cause devices to draw images and texts differently. A fingerprint can exploit this difference to distinguish users. We spoofed the canvas by adding a noise that hides the real canvas fingerprint. To do so, we used the Firefox add-on *Canvas Defender* [13]. To test the add-on efficiency, we inject a canvas fingerprinting script on our own website. The script first draws on the user's browser. Next, the script calls the Canvas API `ToDataURL` method to get the canvas in dataURL format and returns its hashed value. This hashed value can then be used as a fingerprint. To evaluate the add-on efficiency against the canvas fingerprinting, we revisited the customized website and compared the rendered canvas fingerprint. We found that the returned canvas hashed values were different upon every visit.

IP address. We run the *Initial crawl* with an IP address pointing to France. We spoofed the IP address using the VPN add-on *Browsec VPN* [71]. We used a static IP address pointing to the Netherlands. Consequently, the spoofed IP address remains constant during the runs of spoofed crawls. We checked that the IP address changed using the *whoer* website.

Time zone. We launched the *Initial crawl* with *Paris UTC/GMT +1* timezone. We spoofed the timezone to *America/Adak (UTC-10)* using the *Chameleon* add-on [16].

Accept-language. The Accept-language header specifies which languages the user prefer. We used English as Accept-language in *Initial crawl*. We spoofed the Accept-language header using the *Chameleon* add-on [16] to Arabic. We checked that it was properly spoofed using the *whoer* website.

4.5 Limitations

Spoofing features and implementing the spoofing solution with the OpenWPM crawler requires substantial engineering effort. Therefore, we limit our study to 8 browser features that are commonly used by previous works and that can be spoofed either directly using browser settings, or using the add-on (Canvas Defender, Browsec VPN, and Chameleon) that we successfully run with OpenWPM. Consequently, cookies respawned using other features are excluded from this study. The number of excluded cookies is 2,976 (see Section 5.1.1). This is a limitation that does not impact the main goal of our study, as we do not intend to be exhaustive in the identification of respawned cookies, but we aim to understand and describe the mechanisms behind respawning, and propose a robust methodology to detect features that are used by trackers to respawn cookies.

Given that we spoof one feature at a time, we may introduce inconsistency between different features. For example, when we spoof the geolocation API, we do not modify the time zone or the IP address. This method doesn't invalidate our results because we detect dependency on each feature separately. Nevertheless, we may miss trackers that modify their behaviour when some features are spoofed.

Non user-specific cookies are not intrusive for the user's privacy because they are identical among different users. We are aware that the cookies we classify as non user-specific might have been respawned due to features we do not consider.

The usage of a stateless crawler in this study may impact websites' behavior. Zeber et al. [75] showed that stateless crawlers might experience a higher rate of third-party activity compared to real users keeping browser state. This is not surprising as cleaning the browser storages during stateless crawl forces again the creation of third party cookies each time we visit a site. The goal of our work is to study cookies respawning that happens when users are cleaning their browser storage. Therefore, stateless crawlers are best suited to simulate users that are consistently cleaning their browsers.

In this work we use add-ons to spoof features. Such add-ons could be used themselves to fingerprint users. We only use add-ons during the Spoofing crawls. Therefore, all the candidate respawn cookies are collected without any interference with add-ons. The goal of the Spoofing crawl is both to show that a cookie is respawned due to a fingerprinting feature and to identify a feature used for the fingerprinting. The fact that an add-on is used to fingerprint the user does not change our conclusion that the cookie is respawn due to fingerprinting. However, we might incorrectly identify the feature responsible for the respawning if the tracker use the add-on itself to fingerprint the user. Indeed, this feature could be the studied feature, a feature built on the presence of the add-on, or a combination of features. However, this will not impact our conclusions that a cookie is respawn due to fingerprinting.

Crawls	*Initial*	*Re-appearance*	*User specific*	*Feature dependent*
# cookies	428,196	88,470	5,144	1,425
# websites	30,000	18,117	4,093	1,150

Table 2. Cookie respawning with browser fingerprinting is common. We detected 1,425 respawned cookies that appear on 1,150 websites. We define the Initial, Reappearance, User specific crawls and Feature dependent cookies in Section 4.3.

5 Results

In this section, we present findings on prevalence of cookie respawning with browser fingerprinting, identify responsible parties, and analyze on which type of websites respawning occurs more often. Our results are based on Alexa top 30,000 websites where we extracted a total of 428,196 cookies. We study the respawning of both first and third party cookies.

5.1 Overview of cookie respawning with browser fingerprinting behaviour

In this section, we present a general overview of the results of our study. First, we present the prevalence of cookie respawning with browser fingerprinting in the studied 30,000 websites, then we manually analyse a random subset of the respawned cookies.

5.1.1 How common is cookie respawning with browser fingerprinting?

Table 2 presents an overview of the prevalence of cookie respawning with browser fingerprinting. We extracted 428,196 cookies from the visited 30,000 websites. Using the *Reappearance crawl*, we extracted a set of cookies that did reappear in the crawl. As a result, we obtained a set of 88,470 (20.66%) reappearing cookies that appear on 18,117 (60.39%) websites.

Next, we filtered out cookies that are not user-specific – they appear with the same (host, key, value) on *Initial crawl* and *User specific crawl* – and cookies that only appear on *Initial* but not in *User specific crawl* (Section 4.3.1). We found that out of 88,470 reappearing cookies, 5,144 (5.81%) are user specific. The set of user specific cookies is observed on 4,093 (22.59%) websites.

After filtering out non reappearing cookies and keeping only user specific cookies, we identified cookies whose value depend on at least one of the studied features following our methodology detailed in Sections 4.3.2. As a result, we extracted 1,425 respawned cookies that appear on 1,150 (3.83%) websites. Out of the remaining 3,719 cookies, 743 were excluded from the statistical test because they did not appear on the 99 spoofing and control crawls.

The remaining 2,976 cookies that are user specific and not detected as feature dependent can be respawned via other features. Indeed, we remind that in this work we consider 8 fingerprinting feature (see Section 4.3.2), so any other feature could be behind the recreation of the 2,976 cookies. However, the exploration of reasons behind the recreation of these 2,976 cookies is out of scope of our study.

Summary. We found 1,425 cookies respawned using at least one of the studied features. These cookies were respawned in 1,150 websites that represent 3.83% of the visited websites.

5.1.2 Can manual analysis infer the purpose of respawned cookies?

To evaluate the purpose of the respawned cookies, we manually analyzed a 10% random subset of the 1,425 respawned cookies (142 cookies). We searched for the purpose description of the cookies, then, we manually analyzed the cookies values and keys to explore the information stored in these cookies.

Websites, as part of their accountability and transparency obligations, need to declare the purposes of cookies that they use in their websites. In practice, websites describe the purposes of cookies in the corresponding privacy policies (or in cookie policies) [30]. Therefore, to capture the 142 respawned cookie purposes, we searched for cookie policies as follows. If the cookie is set by the visited website, we directly visit the website, and search for the cookie policy. Otherwise, if the cookie is set by a third party domain, we perform a google search for the concatenation of the cookie key, the third party domain name, and the string "cookie policy". We then search for a cookie policy in the 5 first google search results pages.

We found 24 (17%) cookie purpose descriptions in cookie policies, out of which 12 (8.45%) cookie purposes infer the collection of user related information. The remaining cookie purposes do not provide any description on the usage of the cookie. We were not able to analyze a remaining 15 website policies because they were writ-

ten in a language we do not speak. Overall, none of the cookie purposes mentioned the respawning behavior.

To gain insight on the respawned cookies, we analyzed the content of the 142 cookies manually. First, we searched for the 8 studied feature values, we found that 18 cookies included browser and machine features in their value in clear text. We noticed that the remaining 124 respawned cookie values are presented as a sequence of numbers and letters, and that 79% of these cookies have a length ≥ 5. These values refer most likely to the identifier associated to the user, or encrypted/encoded browser and machine features.

The content of a cookie might be inferred from its key. Analyzing the 142 respawned cookie keys, we found that 15 cookies have a key that contains one of the following words associated to a studied feature: GEO, IP, browser, location, UA.

Summary. Out of the 142 cookie policies we accessed, less than 9% refer to the collection of user related information and none of them describe the respawning behavior we observed. Direct observation of cookie keys and values does not bring much more information. Therefore, to understand novel cookie usages, we cannot rely on cookie policies or simple cookie inspection. Sophisticated measurement methodologies must be developed instead to discover new usages. Our work is the first step of this effort.

5.2 Which features are used to respawn cookies?

Our goal is to show a statistical significance of a cookie being dependant on a feature. We do not intend to exhaustively explore all features that are actually used by trackers to respawn cookies. Therefore, when we show that a feature is used to respawn a cookie, it does not mean that this feature is used alone (most likely, features are used in large combination), but that we exhibited a cookie respawning behavior with a fingerprinting technique.

For each of the $1,425$ respawned cookies, we detected features on which the cookie value depends (see all studied fingerprinting features in Table 1).

Table 3 presents the number of times each feature is used to respawn a cookie. IP address is the most commonly used feature to respawn cookies and is used in respawning of 672 (47.15%) cookies. The second most popular feature to respawn cookies is User-Agent (UA) – it is observed with 486 (34.10%) cookies. Note that features that can be easily collected passively, like IP address and UA, are more frequently used than features that can only be accessed actively, such as Canvas.

	Passive	Active/Passive			Active			
Features	**IP**	**UA**	**Lang**	**DNT**	**CV**	**GEO**	**GL**	**TZ**
Occurrence	**672**	**486**	**278**	**277**	**231**	**249**	**292**	**310**

Table 3. IP address is the most commonly used feature to respawn cookies. Occurrence: number of times a feature has been used to respawn a cookie (either independently or in combination with other features). CV: Canvas, IP: IP address, UA: User agent, GEO: Geolocation, GL: WebGL, TZ: Time zone, Lang: Accept language, DNT: Do Not Track.

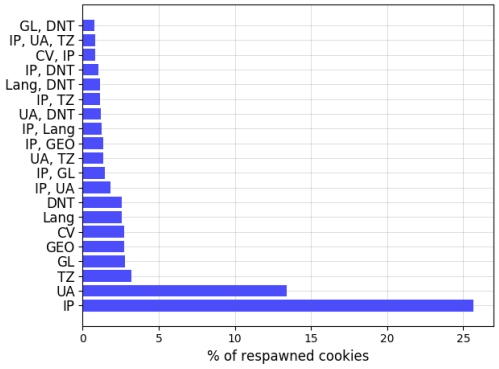


Fig. 3. Top 20 set of features used to respawn cookies. IP addresses alone are used to respawn over 25% of the cookies. CV: Canvas, IP: IP address, UA: User agent, GEO: Geolocation, GL: WebGL, TZ: Time zone, Lang: Accept language, DNT: Do Not Track.

Given that a cookie can be respawned with several features, we consider that a cookie C *is respawned with a set of features* F if the value of C depends on every feature in F (such detection was done independently for each feature as described in Section 4.3.2).

We found that cookies are usually respawned with a set of different fingerprinting features. In our dataset, cookies are respawned with 184 distinct sets of feature combinations. Figure 3 shows the sets of features most often used for cookie respawning. We see that the IP address alone is the most commonly used feature to respawn cookies, and moreover no other set of features is more popular than the IP address alone.

The IP address is used alone to respawn 366 (25.68%) cookies. Mishra et al. [47] studied the stability and uniqueness of the IP address over a duration of 111 days on a dataset of 2,230 users. They showed that 87% of participants retain at least one IP address for more than a month. Hence, IP addresses are both stable and unique, therefore, they can be used to uniquely identify and track user's activity. Interestingly, the top-

2 sets of features, {IP}, and {UA}, contain only passive features that are easier to collect. Active features are rarely used: timezone, the most popular active feature for respawning, is used alone for 46 (3.23%) cookies.

We designed a robust methodology to find a correlation between respawned cookies and fingerprinting features. However, it can be argued that by analyzing the network traffic we can reach the same conclusion. To evaluate this hypothesis, we performed a network traffic analysis on 10% of the websites where cookies are respawned (115 websites). We analyzed both the request URLs, and the POST payloads sent to the tracker on the website. We searched for the presence of the studied fingerprinting feature values in the requests in clear text, hashed with SHA1, or encoded using B64. We found for only 7 (6%) websites that at least one of the studied features value is sent to the tracker. It is clearly a challenge to directly observe such values in the network traffic. Features can be sent in an encoded or encrypted way to the server either alone or in combinations. In addition, the way features are sent can be easily changed by the trackers, but identified with high difficulty by an observer. The methodology we developed in this paper overcome these issues.

Summary. We show that trackers use multiple combinations of features to respawn cookies and that the IP address, which is overlooked in a number of fingerprinting studies [2, 14, 32, 37, 48], is the most used feature to respawn cookies.

5.3 Discovering respawned cookies owners

Due to the the Same Origin Policy (SOP) [57], the domain that is responsible for setting a cookie can be different from the domain that receives it (see Section 2.1). Therefore, we differentiate two stakeholders: *owner* – the domain that is initially responsible for setting the cookie, and *host* – the domain that has access to the cookie and to whom the cookie is sent by the browser via HTTP(S) headers. In the following, we define both owner and host as 2^{nd}-level TLD domains (such as `tracker.com`).

It is important to detect the cookie owner – for instance, in order to block its domain via filter lists [19–21] and prevent cookie-based tracking. Indeed, the notion of cookie owner is often overlooked when the reasoning is only based on the cookie host [12]. When one cookie owner sets a cookie in the context of several websites (the owner's script can be embedded directly on a visited website or in a third-party *iframe*), the host of this owner's cookie is the context where the cookie is set because of the SOP [57]. To identify cookie owners, we distinguish two cases, as described below.

Cookie set by HTTP(S) header. If the cookie is set by the HTTP(S) `Set-Cookie` response header, then the *owner* of the cookie is the same as its *host* because it corresponds to the 2^{nd}-level TLD of the server that sets the cookie.

Cookie set by a script. `Document.cookie` property is the standard way for a JavaScript script to set a cookie [35] programmatically. To check whether a cookie is set via JavaScript and to extract its *owner* (the domain who serves the script) when crawling a website, we (1) extract the set of scripts S that set a cookie on the website using `document.cookie`, (2) for every script in S, we extract the set of cookies C set by this script, and (3) extract the set of respawned cookies identified in Section 5.1.1 that are never set by HTTP(S). Then, we check whether there is an overlap between the (key,value) set of these respawned cookies and the set C. If it is the case, we conclude that the cookies in the overlap are set via JavaScript, and their *owner* is the 2^{nd}-level TLD domain that served the script.

For each of the 1,425 respawned cookies, we identified its *owner* depending on how the cookie was set. Figure 4 shows domains appearing as host only (left blue part), as owner only (yellow part), or both (middle overlap). In total, 1,425 respawned cookies are labeled with 765 distinct hosts, however they were set by 574 distinct owners. Figure 4 also depicts that 75 domains appear as owners and never as cookies hosts. These domains serve JavaScript scripts that set cookies, but never serve cookies directly via an HTTP(S) response header. Hence, when only considering cookies hosts, these domains are not detected. We evaluated the efficiency of Disconnect [19] filter list in detecting these 75 domains. We found that Disconnect miss 53 (70.66%) owners domains. We also found that 266 domains that appear as cookie hosts are never identified as cookie owners. Cookies associated with these domains were set in the context of the hosts domain because of the SOP, but these domains were never actually responsible of setting these cookies.

Figure 5 presents the top 10 domains responsible for cookie respawning that are either cookie hosts, cookie owners, or both. Two domains, `rubiconproject.com` and `casalemedia.com`, represent the largest fraction of websites. All cookies served by these two domains are served via HTTP(S). Three out of the top 10 domains are exclusively cookie owners: `adobetm.com`, `bizible.com`, and `maricopa.gov`. These domains are

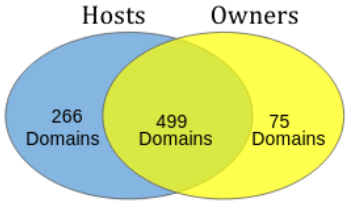


Fig. 4. Emergence of new domains when considering cookie owners. The 1,425 respawned cookies have 765 distinct hosts and 574 distinct owners. The notion of cookie owner allows to identify 75 cookie owner domains that never appear as a cookie host. We also found 266 cookie host domains that are never used to set the cookie.

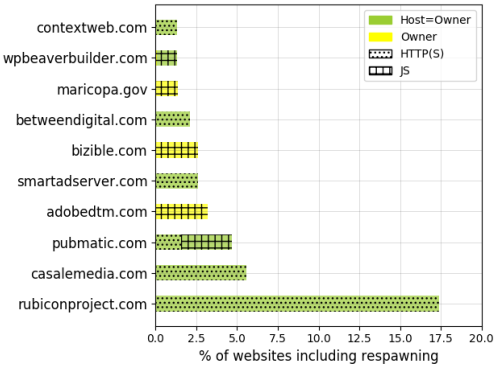


Fig. 5. The top 10 respawned cookies owners. The bar is green when the domain is both host and owner, and yellow when the domain only appears as owner. For each domain, we show when cookies are set via an HTTP(S) header and when they are set via JavaScript. When considering cookie owners, new domains are identified such as `adobedtm.com`.

only setting respawned cookies via JavaScript and never directly through HTTP(S). Out of the 1,425 respawned cookies, 514 (36.07%) are set via JavaScript.

Summary. Previous studies that only looked at the cookie host can miss the trackers responsible for setting the cookies. In our study, 75 domains could be missed if we only considered cookie hosts. We found that Disconnect [19] miss 70.66% of these domains. Considering cookie owners improves the understanding of the tracking ecosystem.

5.4 Where does respawning occur?

In Section 5.3, we studied the domains that are responsible of setting and respawning cookies. In this section, we analyse on which types of websites respawning occurs. In the following, we refer to these websites as *websites including respawning*. We analyse Alexa ranking distribution and impact of websites category on cookie respawning, present websites including respawning that process special categories of data, and present the geolocation of owners of respawned cookies and websites.

Alexa rank interval	Websites including respawning	# of owners
0 — 1k	49 (4.9%)	49
1k — 10k	360 (4%)	213
10k+	741 (3.70%)	382

Table 4. Popular websites are more likely to include cookie respawning. Number of owners: presents the total number of distinct respawned cookies owners in the ranking interval.

Popularity of websites including respawning. We detected 1,150 websites where at least one cookie is respawned. Table 4 presents the number of websites including respawning for each Alexa rank interval. We observe that cookie respawning with browser fingerprinting is heavily used on popular websites: out of the top 1k visited websites, 4.9% are including respawning. This percentage decreases to 3.70% in less popular websites.

Categorization of websites including respawning. We used the *McAfee service* [42] to categorize the visited websites. The McAfee uses various technologies and artificial intelligence techniques, such as link crawlers, and customer logs to categorise websites. It is used by related works [68]. A description of the reported McAfee categories can be found in the McAfee reference guide [43].

We successfully categorized Alexa 29,900 visited websites. We found that the visited websites belong to 669 categories and the 1,150 websites including respawning belong to 143 different categories.

Figure 7 of Appendix gives an overview of the 10 most prominent categories within the Alexa visited websites. We found that all top 10 categories contain websites that include respawning. Business is the top websites category, 8.62% of the visited websites are categorized as business.

Most of websites including respawning are categorized as *General News*. Out of the 29,900 visited websites, 6.73% are categorized as *General News*, and 5.95% of these *General News* websites contain at least one respawned cookie. *General News* is known for using more third parties than other categories [64], which can be the reason behind the high deployment of respawning in this category of websites.

Websites processing special categories of data. The GDPR [66, Recital 51] stipulates that personal data which are particularly sensitive by their na-

ture, merit specific protection, as their processing could create significant risks to the fundamental rights of users. Such data include personal data revealing sensitive information such as data concerning a natural person's sex life or sexual orientation [66, Article 9]. Processing such categories of data is *forbidden*, unless allowed by the user's explicit consent [66, Article 9(2)].

We studied tracking via the third-party respawned cookies on websites processing sensitive data. As a result, we detected 21 cookies respawned in *Adult* websites that are set by 19 different owners. The top domain respawning cookies on sensitive websites is `adtng.com` (no corresponding official website was found for `adtng.com`). It respawned cookies on 3 different adult websites, and therefore, can track and link user's activity within adult websites in a persistent way, *without explicit consent to legitimize such operation*, rendering such respawning practise unlawful.

Geolocation of websites including respawning and respawned cookies owners. Independently of the country of registration of a website, if a website *monitors* the behavior of users while they are in the EU, the GDPR applies to such monitoring [66, Article 3(2)(b)]. Notice that any form of web tracking will be deemed as "monitoring", including cookie respawning with browser fingerprinting. Since our experiments simulate users located in France (EU), both EU and non-EU organizations must comply with the GDPR.

We extracted the country of registration of the owners of respawned cookies and the websites including them using the *whois library* [73]. We successfully identified the country of registration of 362 (63.07%) out of 574 total distinct owners, and 670 (58.26%) out of 1,150 websites including respawning. Out of these 670 websites, 52 (7.76%) are in the EU. We found that the owners and websites are distributed across the globe, ranging respectively over 29 and 47 different countries.

Figure 8 of Appendix presents the registration countries of respawned cookies owners and websites where they are set. We observe that top countries of both respawned cookies and websites including respawning are not in the EU: 356 (24.98%) of the respawned cookies are both originated and included by domains from the US. We also observed that respawned cookies on Chinese websites are only set by Chinese owners, and interestingly, websites registered in Panama are active in respawning as well (22 (3.28%) of the studied 670 websites including respawning are from Panama).

Summary. Cookie respawning with browser fingerprinting is commonly used: 5.95% of *General News* websites contain at least one respawned cookie. We found that cookies are respawned in sensitive *adult* websites as well, which leads to serious privacy implications. Cookie respawning with browser fingerprinting is distributed across the globe, however, only 7.76% of the websites that include respawning are in the EU. Nevertheless, both EU and non-EU websites must comply with the GDPR as it is applicable independently of the country of registration of the website.

5.5 Tracking consequences of respawning

This paper does not specifically focus on cross-site tracking. Indeed, our methodology based on stateless tracking is designed to study *cookie respawning with browser fingerprinting*, not cross-site tracking that would require a stateful crawl, which is beyond the scope of this paper. However, in this section, we describe that cookie respawning with browser fingerprinting is already used today for cross site tracking, including cross-site tracking using first party cookies only.

5.5.1 Respawned cookies purposes

In this section, we evaluate the purpose of the respawned cookies. Given that only a small percentage of cookies includes a description of their purposes [30], we use the Cookiepedia open database [17] that has been used in prior work [68]. It is the largest database of pre-categorized cookies with over 11 million cookies used across 300,000 websites. It uses the classification system developed by "The UK International Chamber of Commerce" (ICC) and relies on four common purposes of cookies: i) Strictly Necessary (which includes authentication and user-security); ii) Performance (also known as analytics, statistics, or measurement cookies); iii) Functionality (includes customization and multimedia content); and iv) Targeting (known as advertising).

We found that out of the $1,425$ respawned cookies, 134 (9%) were not found in the cookipedia database, 955 (67%) were categorized as Unknown, and 336 (24%) were successfully categorized. Out of the 336 cookies, 24% are categorized as targeting cookies (Table 5). We remind that all respawned cookies are user specific thus allow to identify users (Section 4). Therefore, theses cookies could be used for tracking even if it is not their declared or initial purpose.

In the following, we will refer to *potential tracking* as the technical ability to track user's activity using the respawned cookie. However, as described in this section,

Purpose	# of cookies	Purpose	# of cookies
Targeting/Advertising	80	Strictly Necessary	99
Functionality	24	Performance	133

Table 5. Cookipedia classification

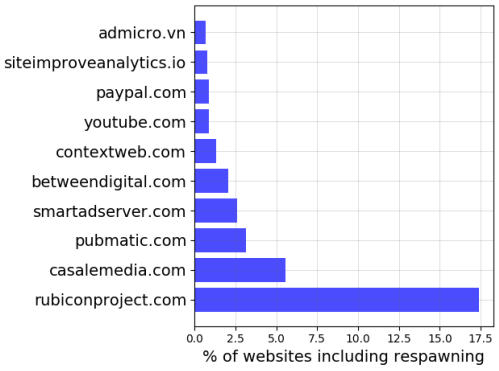


Fig. 6. Persistent third-party tracking based on respawned cookies. Top 10 cross-site trackers using respawned cookies.

companies might use such cookies for a different purpose.

5.5.2 Persistent cross-site tracking with respawned cookies

Basic tracking via third-party cookies [29, 55] is the most known tracking technique that allows third parties to track users across websites, hence to recreate their browsing history. When a third party cookie that enables cross-site tracking is respawned, such tracking becomes *persistent*. That is, in contrast to regular third-party tracking, the user can not prevent it by deleting cookies. Hence, respawned cookies enable persistent potential tracking that allows trackers to create larger users' profiles by linking users activity before and after they clean their browser. Since the host is the domain to whom browser automatically sends the cookies, we focus on the cookie *host* and not on cookie owner.

Third party cookies allow trackers to track users *cross-websites* [55]. In this section, we only analyse third-party respawned cookies that can be used to track users across websites. Note that all extracted respawned cookies are user-specific (Section 4.3.1) and therefore can be considered as *unique identifiers*. Out of 1,425 respawned cookies, 528 (37.05%) are third-party cookies. In total, we identified 144 unique hosts that have access to these cookies.

Figure 6 presents the top 10 cross-site trackers that have access to respawned cookies. We found that `rubiconproject.com` is the top domain: it has access to at least one respawned cookie on 200 (17.39%) of the visited websites. `Rubiconproject.com` defines itself as a publicly traded company, as it is automating the buying and selling of advertising [56].

5.5.3 Cookie respawning with browser fingerprinting beyond deprecation of third-party cookies

Web browsers are moving towards the deprecation of third party cookies which are the core of cross-site tracking [58]. *Can this deprecation prevent cross-site tracking?* In the following, we show how cookie respawning with browser fingerprinting can overcome browsers preventions.

Via *persistent tracking with respawned cookies*, domains can track users across websites without third-party cookies. Consider the following scenario: `example.com` and `news.com` include a fingerprinting script from `tracker.com`. When the user visits these websites, the script from `tracker.com` accesses the user's browser and machine features, and sets a corresponding first-party cookie. As a result, two first-party cookies are set in the user's browser and labeled with two different hosts: `example.com` and `news.com`, but the values of these two cookies are identical, because they are created from the user's browser and machine features. By respawning these two cookies on both websites, the owner `tracker.com` shows to be able to track the user in a persistent way across sites with a first-party cookie only.

We analyzed the usage of the same (owner, key, value) first-party respawned cookie across different websites. The 1,425 cookies correspond to 1,244 respawned (owner, key, value) instances, out of which 40 (3.21%) are respawned on multiple websites in a first party context with the same value (see Table 9 in Appendix). `wpbeaverbuilder.com` [74] is the top owner setting identical first party respawned cookies across websites. It respawned the same cookie on 15 distinct websites. It defines itself as a WordPress page builder. Its policy declare to collect user's information, but it does not specify the type of this information. We found that `google-analytics.com` respawned the same cookie on 7 websites owned by WordPress. We suspect that the respawning on these websites is a result of WordPress configuration of the google-analytics service.

Summary. Cookie respawning with browser fingerprinting enable potential tracking across websites even when third party cookies are deprecated. We found 40 first party cookies that can serve for cross-site tracking.

6 Is respawning legal?

In this section, with a legal expert co-author of the paper, we evaluate the legal compliance of 1,425 respawned cookies and the reasons why this practice is unlawful, regardless of the purposes for which cookies are respawned, as it infringes core GDPR principles. Our legal analysis is based on the General Data Protection Regulation (GDPR) [66] and the ePrivacy Directive (ePD) [26], as well as in its recitals. The GDPR applies to the processing of personal data [28] and requires that companies need to choose a legal basis to lawfully process personal data (Article 6(1)(a)). The ePD provides *supplementary* rules to the GDPR in particular in the electronic communication sector, such as websites. We have additionally consulted the guidelines of both the European Data Protection Board (an EU advisory board on data protection, representing the regulators of each EU member state) [1] and the European Data Protection Supervisor (EDPS, the EU's independent data protection institution) [23]. While these guidelines are not enforceable, they are part of the EU framework for data protection which we apply in this work to discern whether respawning is compliant.

To assess the legal consequences of respawning, the legal expert analysed legal sources to interpret cookie deletion. To our surprise, we found that there is *no explicit legal interpretation of cookie deletion.* Only the EDPS [23, Section 4.3.4] noted that *"if cookies requiring consent have disappeared, this is most probably because the user deleted them and wanted to withdraw consent"*. As a result, *cookie respawning also does not have a clear legal interpretation* and merits attention for its plausible legal consequences. These consequences can arguably be derived, not only from the consent perspective, but also from the core principles of data protection, as discussed in the following sections (fairness, transparency and lawfulness principles). Thus, owners of respawned cookies and website owners that embed those may be jointly responsible for their usage (Article 26 [66]) and may then be subject to fines of up to 20 million EUR (or 4% of the total worldwide annual turnover of the preceding financial year, Article 83(5)[66]).

6.1 Fairness principle

This principle requires personal data to be processed fairly (Article 5(1)(a)). It requires that i) *legitimate expectations of users are respected* at the time and context of data collection, and ii) there are no "surprising effects" or potential negative consequences occurring in the processing of user's data.

Findings: We consider that *all* 1,425 *respawned cookies plausibly violate the fairness principle*, as respawning seems to be inconsistent with the user's expectations regarding respawned cookies after its deletion from their browser, and also considering the cookie's duration.

Suggestions for policymakers: It is hard to operationalize the high-level fairness principle into concrete requirements for website owners and map it into legitimate expectations of users. Policy makers need to provide more concrete guidelines on the operationalization of this principle in the Web.

6.2 Transparency principle

Personal data processing must be handled in a transparent manner in relation to the user (Article 5(1)(a)). This principle presents certain obligations for websites: i) inform about the *scope and consequences* [67] and the *risks* in relation to the processing of personal data (Recital 39); ii) inform about the purposes, legal basis, etc. before processing starts (as listed in Art. 13); iii) provide the above information in a concise, transparent, intelligible and easily accessible form (Art. 12).

Findings: We analyzed the privacy policies of the 10 top popular respawned cookie owners: *rubiconproject.com [56], casalemedia.com [15], pubmatic.com [53], adobedtm.com [4] smartadserver.com [61], bizible.com [10], betweendigital.com [9], maricopa.gov [41], wpbeaverbuilder.com [74], and contextweb.com* (Figure 5). Some policies [4, 10, 15, 56, 61] refer to the use of browser's features without referencing the consequences or risks thereof. Also, none of the policies refer to cookie respawning. As such, these seem to be in breach of the transparency principle.

Suggestions for policymakers: In practice, the description of data (purposes, legal basis, types of personal data collected, features used and its consequences) is often mixed within the text, which makes harder to extract concrete information therefrom [30]. Policy-makers need to converge on harmonized requirements and standard format for privacy policies.

	Session	Persistent
First-party	Targeting/ Advertising	Targeting/ Advertising
Third-party	Targeting/ Advertising Performance Strictly necessary	Targeting/ Advertising Performance Strictly necessary Functionality

Table 6. Purposes of Cookiepedia [17] that require consent according to their context and duration.

6.3 Lawfulness principle

The ePD requires websites to obtain user consent to lawfully process personal data using cookies. When a cookie recreates itself without consent, every data processed henceforth could be considered unlawful due to lack of legal basis for personal data processing [27]. Hence, cookie respawning incurs in violation with the lawfulness principle (Articles 5(1)(a) and 6(1) of the GDPR, and 5(3) of the ePD). The EDPS [23] already advised against the use of cookie respawning if the processing relies on users' consent. It mentions that *"cookie respawning would circumvent the user's will. In this case (...) institutions must collect again user's consent"*.

To evaluate compliance with the lawfulness principle, we need first to evaluate whether cookies are exempted or subject to consent. Pursuant to it, it is determinant to identify the *purpose* of each cookie, since *"it is the purpose that must be used to determine whether or not a cookie can be exempted from consent"* [7].

Two other characteristics contribute to determine whether cookies are exempted or subject to consent [7]: duration (*session and persistent cookies*) and context (*first and third-party cookies*). Building on the analysis of Santos et al. [59, Table 5] on the *purposes* that are subject to consent, we studied the Cookiepedia purposes, and then derived those that are subject to consent according to their duration and context. Table 6 summarizes the Cookiepedia purposes requiring consent depending on their duration and context.

Findings: In our study we crawled websites and even if a website provided a consent banner, we did not give consent thereto. We evaluated whether respawned cookies are subject to or exempted from consent (as described in Table 6). As a result of our evaluation, we found that out of 336 respawned cookies categorized by Cookiepedia, 130 (38.69%) are subject to consent. Hence, these 130 cookies are in breach of the lawfulness principle.

Suggestions for policymakers: Companies can embed respawning and still claim respawned cookies are exempted of consent. We analysed that both the duration and context of cookies contribute to determine whether cookies are exempted or subject to consent. However, from a technical point of view, these criteria can be bypassed by domains that embed respawning. As per *duration*, session cookies can get recreated even after their elimination by the user. Functionality cookies are exempted of consent when used as session cookies and are subject to consent when used in a persistent way [7]. When respawned, such cookies can be used for a longer duration than previously envisaged. We found that out of 1,425 respawned cookies, 446 (31.30%) are session cookies. Regarding *context*, performance cookies are exempted of consent when used in a first party context and are subject to consent when used as third party cookies. However, in practice, a cookie set in the first party context can be considered as a third party cookie in a context of a different website. We found that 4 respawned cookies (host,key,value) appear as first- and third-party in different websites. These cookies are respectively set by `pornhub.com`, `mheducation.com`, `hujiang.com` and `fandom.com`. Given that a cookie context and duration can be altered, these should not be used as a criteria to evaluate the need of consent.

7 Conclusion

This work presents a large scale study of cookie respawning with browser fingerprinting, a tracking technique that is devoid of a clear legal interpretation in the EU legal framework. We employed a novel methodology to reveal the prevalence of cookie respawning with browser fingerprinting in the wild. The detection of such behavior and the identification of responsible domains can prove to be hard to achieve, which impacts both the ability to block such behavior, and its legal assessment. We believe this work can serve as a foundation for improvement of future regulation and protection mechanisms.

Acknowledgments

This work has been supported by the ANSWER project PIA FSN2 No. P159564-2661789/ DOS0060094 between Inria and Qwant. This research has also been partially funded by the Hauts-de-France region in the context of the ASCOT project of the STaRS framework.

References

[1] 29 Working Party. https://edpb.europa.eu/our-work-tools/article-29-working-party_en.

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

[3] Gunes Acar, Marc Juárez, Nick Nikiforakis, Claudia Díaz, Seda F. Gürses, Frank Piessens, and Bart Preneel. Fpdetective: dusting the web for fingerprinters. In *2013 ACM SIGSAC Conference on Computer and Communications Security (CCS'13)*, pages 1129–1140, 2013.

[4] Adobedtm.com privacy policy. https://www.adobe.com/privacy/policy.html.

[5] 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 A. 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.

[6] Alexa websites. https://www.dropbox.com/sh/wnmugbzb2bfp7ca/AACUJbCbFM2iBcN7y2b-bqF-a?dl=0.

[7] Article 29 Working Party. Opinion 04/2012 on Cookie Consent Exemption (WP 194).

[8] Mika D Ayenson, Dietrich James Wambach, Ashkan Soltani, Nathan Good, and Chris Jay Hoofnagle. Flash cookies and privacy ii: Now with html5 and etag respawning. Technical report, Available at SSRN: https://ssrn.com/abstract=1898390orhttp://dx.doi.org/10.2139/ssrn.1898390, 2011.

[9] Betweendigital.Com privacy policy. https://betweenx.com/.

[10] Bizible.com privacy policy. https://documents.marketo.com/legal/privacy.

[11] Károly Boda, Ádám Máté Földes, Gábor György Gulyás, and Sándor Imre. User tracking on the web via cross-browser fingerprinting. In *16th Nordic Conference on Secure IT Systems, NordSec 2011*, pages 31–46, 2011.

[12] Aaron Cahn, Scott Alfeld, Paul Barford, and S. Muthukrishnan. An empirical study of web cookies. In Jacqueline Bourdeau, Jim Hendler, Roger Nkambou, Ian Horrocks, and Ben Y. Zhao, editors, *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 891–901. ACM, 2016.

[13] Canvas defender. https://addons.mozilla.org/en-US/firefox/addon/no-canvas-fingerprinting/.

[14] Yinzhi Cao, Song Li, and Erik Wijmans. (cross-)browser fingerprinting via os and hardware level features. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, 26 February - 1 March, 2017*, 2017.

[15] Casalmedia privacy policy . https://sugru.com/cookies.

[16] Chameleon extension. https://addons.mozilla.org/en-US/firefox/addon/chameleon-ext/.

[17] Cookiepedia Official website. https://cookiepedia.co.uk/.

[18] Anupam Das, Gunes Acar, Nikita Borisov, and Amogh Pradeep. The web's sixth sense: A study of scripts accessing smartphone sensors. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1515–1532. ACM, 2018.

[19] Disconnect Official website. https://disconnect.me/.

[20] EasyList filter lists. https://easylist.to/.

[21] EasyPrivacy filter lists. https://easylist.to/easylist/easyprivacy.txt.

[22] Peter Eckersley. How Unique is Your Web Browser? In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, PETS'10, pages 1–18. Springer-Verlag, 2010.

[23] Guidelines on the protection of personal data processed through web services provided by EU institutions, November, 2016. https://edps.europa.eu/sites/edp/files/publication/16-11-07_guidelines_web_services_en.pdf.

[24] 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 ACM CCS*, pages 1388–1401, 2016.

[25] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W. Felten. Cookies that give you away: The surveillance implications of web tracking. In *Proceedings of WWW 2015*, pages 289–299, 2015.

[26] Directive 2009/136/EC of the European Parliament and of the Council of 25 November 2009. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32009L0136, accessed on 2019.10.31.

[27] European Data Protection Board. Opinion 15/2011 on the definition of consent (WP 187), adopted on 13 july 2011. https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2011/wp187_en.pdf.

[28] European Data Protection Board. Opinion 4/2007 on the concept of personal data (WP 136), adopted on 20.06.2007. https://ec.europa.eu/justice/article-29/documentation/opinionrecommendation/files/2007/wp136_en.pdf.

[29] Imane Fouad, Nataliia Bielova, Arnaud Legout, and Natasa Sarafijanovic-Djukic. Missed by filter lists: Detecting unknown third-party trackers with invisible pixels. volume 2020, 2020.

[30] Imane Fouad, Cristiana Santos, Feras Al Kassar, Nataliia Bielova, and Stefano Calzavara. On Compliance of Cookie Purposes with the Purpose Specification Principle. In *IWPE 2020 - International Workshop on Privacy Engineering*, pages 1–8, Genova, Italy, September 2020.

[31] Ghostery Official website. https://www.ghostery.com/.

[32] Alejandro Gómez-Boix, Pierre Laperdrix, and Benoit Baudry. Hiding in the Crowd: an Analysis of the Effectiveness of Browser Fingerprinting at Large Scale. In *WWW2018 - TheWebConf 2018 : 27th International World Wide Web Conference*, pages 1–10, Lyon, France, April 2018.

[33] Umar Iqbal, Steven Englehardt, and Zubair Shafiq. Fingerprinting the fingerprinters: Learning to detect browser fingerprinting behaviors. In *IEEE Symposium on Security & Privacy*, 2021.

[34] Intelligent Tracking Prevention. https://webkit.org/blog/10218/full-third-party-cookie-blocking-and-more/.

[35] JS cookies. https://www.w3schools.com/js/js_cookies.asp.

[36] Pierre Laperdrix, Nataliia Bielova, Benoit Baudry, and Gildas Avoine. Browser fingerprinting: A survey. *ACM Transactions on the Web (TWEB)*, 14(2):8:1–8:33, 2020. https://dl.acm.org/doi/10.1145/3386040.

[37] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. Beauty and the Beast: Diverting modern web browsers to build unique browser fingerprints. In *37th IEEE Symposium on Security and Privacy (S&P 2016)*, 2016.

[38] W. Davis. KISSmetrics Finalizes Supercookies Settlement. http://www.mediapost.com/publications/article/191409/kissmetrics-finalizes-supercookies-settlement.html,2013..

[39] R. Singel. Online TrackingFirm Settles Suit Over Undeletable Cookies. http://www.wired.com/2010/12/zombie-cookie-settlement/.

[40] 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)*. USENIX Association, 2016.

[41] Maricopa.gov privacy policy. https://www.maricopacountyparks.net/privacysecurity-policies/.

[42] McAfee categorization service. https://www.trustedsource.org/.

[43] Description of McAfee categories. https://www.trustedsource.org/download/ts_wd_reference_guide.pdf.

[44] Using http cookies, MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies#define_where_cookies_are_sent.

[45] N. Mohamed. http://www.wired.com/2009/08/you-deleted-your-cookies-think-again/.

[46] J. Leyden. Sites pulling sneaky flash cookie-snoop. http://www.theregister.co.uk/2009/08/19/flashcookies/.

[47] Vikas Mishra, Pierre Laperdrix, Antoine Vastel, Walter Rudametkin, Romain Rouvoy, and Martin Lopatka. Don't count me out: On the relevance of IP address in the tracking ecosystem. In Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen, editors, *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pages 808–815. ACM / IW3C2, 2020.

[48] Keaton Mowery and Hovav Shacham. Pixel perfect: Fingerprinting canvas in HTML5. In Matt Fredrikson, editor, *Proceedings of W2SP 2012*. IEEE Computer Society, May 2012.

[49] Firefox blocking tracking cookies. https://blog.mozilla.org/blog/2019/09/03/todays-firefox-blocks-third-party-tracking-cookies-and-cryptomining-by-default/.

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

[51] Lukasz Olejnik, Minh-Dung Tran, and Claude Castelluccia. Selling off user privacy at auction. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*, 2014.

[52] Information stored by openwpm. https://github.com/mozilla/OpenWPM.

[53] Pubmatic privacy policy . https://pubmatic.com/legal/platform-cookie-policy/.

[54] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, Christian Kreibich, and Phillipa Gill. Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. In *Network and Distributed System Security Symposium, NDSS*, 2018.

[55] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. Detecting and defending against third-party tracking on the web. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2012*, pages 155–168, 2012.

[56] Rubicon privacy policy . https://rubiconproject.com/rubicon-project-advertising-technology-privacy-policy/platform-cookie-statement/.

[57] Same Origin Policy. https://www.w3.org/Security/wiki/Same_Origin_Policy.

[58] Google Privacy Sandbox. https://www.chromium.org/Home/chromium-privacy/privacy-sandbox.

[59] Cristiana Santos, Nataliia Bielova, and Célestin Matte. Are cookie banners indeed compliant with the law? deciphering EU legal requirements on consent and technical means to verify compliance of cookie banners. *Technology and Regulation*, pages 91–135, 2020.

[60] You should install two browsers. http://www.compukiss.com/internet-and-security/you-should-install-two-browsers.html.

[61] Smartadserver privacy policy . https://www.sublime.xyz/en/legal-mentions.

[62] Konstantinos Solomos, John Kristoff, Chris Kanich, and Jason Polakis. Tales of favicons and caches: Persistent tracking in modern browsers. In *NDSS*, 2021.

[63] Ashkan Soltani, Shannon Canty, Quentin Mayo, Lauren Thomas, and Chris Jay Hoofnagle. Flash cookies and privacy. In *AAAI Spring Symposium: Intelligent Information Privacy Management*, 2010.

[64] Jannick Kirk Sørensen and Sokol Kosta. Before and after GDPR: the changes in third party presence at public and private european websites. In Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia, editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 1590–1600. ACM, 2019.

[65] Ove Sörensen. Zombie-cookies: Case studies and mitigation. In *8th International Conference for Internet Technology and Secured Transactions, ICITST 2013, London, United Kingdom, December 9-12, 2013*, pages 321–326. IEEE, 2013.

[66] The European Parliament and the Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), 2016.

[67] "Guidelines on transparency under Regulation 2016/679, WP260 rev.01. https://ec.europa.eu/newsroom/article29/item-detail.cfm?item_id=622227.

[68] Tobias Urban, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. Beyond the front page: Measuring third party dynamics in the field. In Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen, editors, *WWW '20: The Web*

Conference 2020, Taipei, Taiwan, April 20-24, 2020, pages 1275–1286. ACM / IW3C2, 2020.

[69] Securing your web browser. https://www.us-cert.gov/publications/securing-your-web-browser.

[70] Antoine Vastel, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. FP-STALKER: tracking browser fingerprint evolutions. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 728–741. IEEE Computer Society, 2018.

[71] Browsec vpn. https://addons.mozilla.org/en-US/firefox/addon/browsec/.

[72] Whoer website. https://whoer.net.

[73] Whois library. https://pypi.org/project/whois/.

[74] wpbeaverbuilder.com privacy policy. https://www.wpbeaverbuilder.com/privacy-policy/.

[75] David Zeber, Sarah Bird, Camila Oliveira, Walter Rudametkin, Ilana Segall, Fredrik Wollsén, and Martin Lopatka. The Representativeness of Automated Web Crawls as a Surrogate for Human Browsing. In *Proceedings of The Web Conference 2020*, pages 167–178, 2020.

A Machines characteristics

Table 7 presents the characteristics of machine A and machine B used in our study.

Characteristics	Machine A	Machine B
Date of the crawl	March 2021	March 2021
OS	Fedora 25	Fedora 31
Firefox version	68.0	45.0.1
Location	France	France
IP address	193.51.X.X	138.96.Y.Y
OpenWPM version	v0.9.0	v0.7.0
Language	English (en_US)	German (de_DE)
Time zone	CET	AKST
Geolocation	France	Alaska
Do not track	Null	True

Table 7. Crawls Characteristics. All crawls were performed from machine A except *User specific crawl* that was done from machine B.

B Collected data

We extract the following from the information automatically collected by OpenWPM:

1. For each HTTP request: the requested URL, the HTTP header.
2. For each HTTP response: the response URL, the HTTP status code, the HTTP header.
3. All JavaScript method calls described in Table 8.
4. All cookies set both by JavaScript and via HTTP Responses. On these collected cookies, we perform the following filtering as shown in Figure 2: first, we select cookies recreated after cleaning the cookies database; second, we filter out cookies that are not user-specific; finally, we filter out cookies that are not respawn with studied features (Section 4.3).

JavaScript calls	API
HTML5 Canvas	HTMLCanvasElement, CanvasRenderingContext2D
HTML5 WebRTC	RTCPeerConntection
HTML5 Audio	AudioContext
Plugin access	Navigator.plugins
MIMEType access	Navigator.mimeTypes
Navigator properties	window.navigator
Window properties	Window.screen, Window.Storage, window.localStorage, window.sessionStorage, and window.name

Table 8. Recorded JavaScript calls.

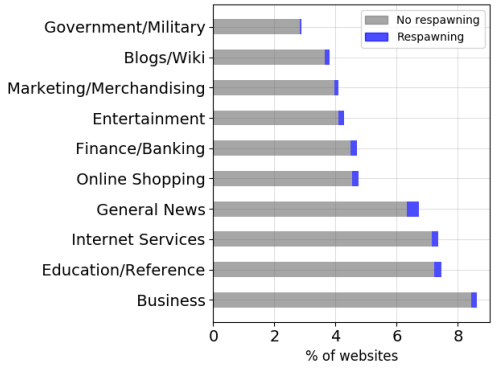


Fig. 7. General news is the top category including cookie respawning with browser fingerprinting. We consider that a website U includes respawning if it contains at least one respawned cookie. The bar is gray when we do not detect respawning in the website, and is blue when we do.

C Additional results

Table 9 presents the top first-party cookies respawned across websites. This practice is studied in Section 5.5.3.

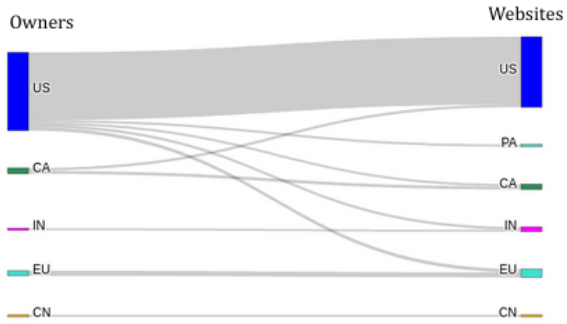


Fig. 8. Cookie respawning with browser fingerprinting is geolocaly distributed. Corresponding countries of owners (left) and websites including respawning (right) of respawned cookies. We present the top 10 (owner,website) geolocation. "EU" label represents the 27 member states of the EU.

Owner	Occurrence
`wpbeaverbuilder.com`	**15**
`clarip.com`	**13**
`maricopa.gov`	**9**
`google-analytics.com`	**7**

Table 9. Top first-party cookies respawned across websites. Every line in the table represents a cookie, hence the same owner can appear on multiple lines. Occurrence: presents the number of websites where the instance (owner,key,value) was respawned.