POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Methodology and Measurements of Privacy Mechanisms for Online Advertising: The Case of Google's Topics API

Advisors

Prof. Marco MELLIA

Prof. Martino TREVISAN

Dr. Nikhil JHA

Candidate

Alberto VERNA

October, 2024

Acknowledgements

Un sincero ringraziamento va ai miei relatori, prof. Marco Mellia, prof. Martino Trevisan e Nikhil Jha, per la loro costante disponibilità e gentilezza. Il loro prezioso supporto ha reso questo progetto di tesi un'esperienza arricchente e stimolante, alleviando anche le difficoltà della stesura.

Un enorme grazie va alla mia famiglia, in particolare a papà Claudio, per il sostegno incondizionato e per avermi permesso di coltivare la mia passione durante il mio intero percorso universitario. È anche grazie al vostro amore, fiducia e incoraggiamento, specialmente nei momenti più difficili, che ora siamo qui a festeggiare questo traguardo.

Abstract

In recent years, the Web industry has been moving towards the abandonment of third-party cookies in favour of more privacy-oriented solutions for targeted online advertising. Among the proposed alternatives, Google's Topics API – a core component of the Privacy Sandbox framework – stands above the rest. It is a browser-based solution for providing a user's topics of interest to a third-party service (e.g. a digital advertising platform) without revealing the websites they visit. As the initial experimentation phase concludes, all components of the Privacy Sandbox, including the Topics API, have now reached general availability. However, some scepticism remains among researchers and privacy advocates who caution that, despite being a better approach than third-party cookies, this new solution may still lead to re-identification attacks and other privacy leaks. For this reason, Google is currently limiting the usage of the Privacy Sandbox family to select third-party services that must undergo an enrolment process. This thesis aims to quantify the adoption of the Topics API within the Web ecosystem, identify third parties that enable its usage, and examine the practices they employ. Given the handling of personal user information, said practices must also be taken in compliance with local privacy regulations.

The analysis was performed by deploying a Chromium-based web crawler to visit the most popular 50,000 websites worldwide, recording the usages of the Topics API on each website. Because the crawling was performed in the EU, where the GDPR is in place, the crawler mimics the behaviour of a user accepting the privacy policy, by finding and clicking the "Accept" button of any banner found inside the visited page. This approach allows to distinguish the usages before and after user consent is provided, only in those websites where a privacy banner is found.

The results of the crawling show that a substantial number of third parties are already experimenting with the new features offered by the Topics API, in preparation of the future phase-out of third-party cookies. However, it's evident that there is still some form of A/B testing taking place on a restricted amount of websites and users, most likely by the third parties. The same results show a significant number of questionable or even anomalous usages: some websites witness Topics API calls that (i) occur before the user accepts the privacy banner, even on European domains, or (ii) come from a third party that has not yet undergone the enrolment process, although the browser is expected to deny calls from unauthorised domains. This latter issue occurs due to an implementation bug found in Chromium's source code, which allows to bypass the browser's authorisation checks by manually deleting or corrupting a specific configuration file. Moreover, the majority of unauthorised domains appear to be same websites

visited by the crawler, hinting at some popular JavaScript libraries accessing the Topics API erroneously.

While this new technology has the potential to replace third-party cookies as the *de facto* standard for interest-based advertising, it is still in its early phases of deployment. The crawling results highlight several issues that are typically associated with early implementation: privacy regulation violations, implementation bugs that allow circumvention of abuse protections, and deployment errors.

Table of Contents

List of Figures

Acronyms

IBA	Interest-Based Advertising
GDPR	General Data Protection Regulation
CCPA	California's Consumer Privacy Act
LGPD	Lei Geral de Proteção de Dados
CMP	Consent Management Platform
FLoC	Federated Learning of Cohorts
FQDN	Fully Qualified Domain Name
CP	Calling Party
GTM	Google Tag Manager
TLD	Top-Level Domain

2LD

Second-Level Domain

Chapter 1

Introduction

1.1 Context

1.1.1 Introduction to online advertising

The rise of online advertising has forever changed the global landscape of promotional content, redefining how businesses and organizations communicate with their audience and allowing companies to reach consumers with never-before-seen levels of scale. Online advertising, compared to its traditional counterpart, leverages the Internet as a mean of displaying advertisements to online users, where *advertisers*—companies wishing to promote their own products—partner with *publishers*, usually websites and other digital platforms willing to host ads inside their own page in exchange for revenue. The rapid growth of online advertising has given way to the birth of full-fledged *advertisement networks*, with the primary goal of interconnecting advertisers and publishers. One of the main challenges for these platforms is matching advertisers and publishers effectively, making sure that the right advertisements are provided to the right people who would be interested in buying the promoted products. The efforts to find more effective advertising strategies led to IBA (Interest-Based Advertising), where the advertisement networks and platforms take the consumer's interests into consideration when delivering ads, thereby personalising the user's ad-viewing experience while also increasing the odds that they buy the promoted products. A relatively simple way to infer an Internet user's interests is to observe what they browse on the Web and how they do it. This is done through *online tracking*, where platforms actively collect information about the users' browsing habits.

1.1.2 Third-party cookies and privacy concerns

For many years, online tracking was accomplished through the use of third-party cookies — small chunks of text that are stored locally in the client's browser, created by a domain different from the one that the user is currently visiting. These domains are usually linked to advertisers, embedded within the publishers' pages. However, the use of cookies has raised significant privacy-related concerns from users, as they potentially allow an advertiser to reconstruct one's browsing behaviour with worrying levels of detail [1, 2, 3]. The central issue lies in the ability of advertising platforms to control the data being stored inside a third-party cookie, which may include sensitive information such as the specific pages a user visits. If present in a large amount of websites, an advertiser can leverage this information to partially reconstruct a user's browsing history. In response to these concerns, several countermeasures to online tracking have been introduced, such as tracker blockers [4, 5, 6], privacy-friendly browsers [7] and search engines [8]. Additionally, legislators have introduced new laws which mandate platforms to obtain user consent before selling or collecting any of their personal information. The most widely recognised of these are the EU's GDPR (General Data Protection Regulation) [9], California's CCPA (California's Consumer Privacy Act) [10] and Brazil's LGPD (Lei Geral de Proteção de Dados) [11]. As a result, companies such as Mozilla, Apple and Google were encouraged to take measures against the usage of third-party cookies on their respective browsers — Firefox, Safari and Chrome. The former two are already blocking them by default [12, 13], whereas Google remains hesitant towards their full deprecation. Despite beginning a phase-out process, whose completion has undergone several postponings [14, 15], the company's current standing on third-party cookies is towards preserving them, while offering users with a more informed choice about their usage [16].

1.1.3 Shifting away from third-party cookies: FLoC and the Topics API

The threat to third-party cookies' existence posed a challenge for the online advertising ecosystem, forcing companies to look for alternative paradigms. Google, a major player in the field, has proposed a collection of privacy-preserving tools that are included directly within the browser, known as the Privacy Sandbox. This framework was introduced to create a healthier ecosystem for online advertising, by offering platforms useful tools for effectively displaying ads inside Web pages while respecting the users' privacy. One of its main components is designed to help advertisers understand the consumers' interests, while keeping their privacy intact: the FLoC (Federated Learning of Cohorts) was Google's first attempt at it, which classified the users into different groups based on their interests using tools directly embedded within the browser. However, following concerns that the rising technology could potentially be exploited to identify users [17, 18, 19], Google abandoned its development and proposed its successor: the Topics API, which will be the main focus of this thesis.

This component takes advantage of the browser's local history to extract the websites the user visits during a certain time interval (at the moment of writing, one week) and maps them into a collection of topics, effectively building a dynamic profile for the user based on their interests. The browser then discloses a controlled subset of topics to authorised third parties (e.g., advertising platforms) that request it. This process is done completely within the browser, so that no data other than the intended topics can be disclosed to third parties. Overall, the Topics API allows advertisers to learn some of the users' topics of interest, thereby helping them improve the efficiency of their advertising, while preventing sensitive information such as their browsing history from being disclosed.

Although it represents an improvement over third-party cookies and the FLoC, the Topics API is still viewed with scepticism from researchers and privacy advocates, as previous studies have shown that user identification remains a viable exploit [20].

1.2 Motivations and methodology

The Topics API, much like the other components of the Privacy Sandbox, has reached general availability and, at the moment of writing, is available to at least 99% of Chrome users [21, 22], thus encouraging online advertisement companies to start experimenting with this new technology. As this delicate process of experimentation takes place, an important aspect is to determine what is the current state of its deployment in real-world applications, including what third parties are using it, how often it is invoked across websites, and whether its adoption is increasing over time. Additionally, given its role in collecting user sensitive information, it is also important to determine whether third parties are using it legitimately and in compliance with existing privacy laws and regulations. This thesis addresses these aspects by painting a global picture of the Topics API's usage in the wild.

We achieve this through an extended measurement campaign that employs a Web crawler based on a headless browser, built for a previous work of research [23]. This crawler is adapted in order to collect extended information regarding the Topics API's usages, recorded from a custom-built version of the Chromium browser. We visit the top 50,000 most popular websites and present results showing which entities are currently experimenting with this new technology. These results will also highlight any instances of unexpected behaviour, typical of technologies in their early stages of production, ranging from inconsistent deployments and questionable integration with privacy regulations to outright erroneous implementations.

Moreover, this thesis work aims to raise awareness among practitioners about the implementation of this new technology and its implications. In fact, the findings in the following chapters will demonstrate the presence of several issues that derive from incorrect configuration, which can be addressed rather easily with a minimal understanding of the Topics API's operation.

1.3 Thesis structure

This thesis is structured as follows:

- Chapter 2 provides an exhaustive explanation of previous ad-related technologies and their issues, addressed by the Topics API.
- Chapter 3 focuses on the Topics API itself, detailing its algorithm, data structures and internal mechanisms that allow the building and disclosure of user profiles based on their topics of interest.
- Chapter 4 concentrates on how the data was collected, detailing the implementation of the Web crawler employed for the analysis and the modifications done to it and to the Chromium browser in order to collect extensive usage information.
- Chapter 5 illustrates the results obtained during the first crawling iteration, giving a global picture of the Topics API usage within the Web ecosystem and highlighting its evolution over an extended period of time. This chapter will also identify some questionable and outright anomalous behaviour exhibited from the browser during the crawling.
- Chapter 6 serves as a conclusion, concentrating on possible directions for future research.

Chapter 2

Background

2.1 Third-party cookies

In the context of Internet browsing, *cookies* are defined as small chunks of text that are created by a Web server and stored locally in the device of a user visiting a Web page with a browser. They were originally introduced in 1994 to address the inherent *statelessness* of the HTTP protocol. Statelessness refers to the fact that HTTP, by design, does not remember any information exchanged in previous interactions between a client (browser) and a server: each request sent to the server is independent, with no built-in way for the server to track session information and user interactions. This limitation posed challenges for scenarios requiring persistent data, such as user authentication, shopping carts and settings that span across multiple pages. To solve this problem, cookies were introduced as a mechanism to store information about the state of an HTTP interaction within the user's device. This information can be then sent back to the server with every subsequent interaction, allowing the server to "remember" it across multiple requests. A server can create a cookie by commanding the browser to store it using the `Set-Cookie` header in its HTTP response. Similarly, the client can communicate its stored cookies to the server using the `Cookie` header in its requests. Nowadays, cookies are primarily used to store data related to the user's authentication state, site-wide settings and interactions with a website, improving their overall browsing experience.

Cookies can be classified based on their persistence:

- *Session cookies* only last for the duration of a browser session and are deleted when the user closes the browser.

- *Persistent cookies* remain across multiple browser sessions, so they are not deleted when the user closes the Web browser. A common use case is the storage of a user's access token in a persistent way, so that they can remain logged in even after closing the browser.

Cookies can also be classified based on the domain of the Web server that creates them:

- *First-party cookies* are directly created by the website that the user is visiting.
- *Third-party cookies* are created by a different domain from the one the user is visiting.

Persistent third-party cookies have been the primary solution used by advertising platforms to store and track information about the users' browsing habits. When a user visits a web page that displays ads, their browser performs an HTTP exchange with an advertising service—a *third party*. During this exchange, the advertiser can instruct the browser to create a third-party cookie containing information relevant to the user's browsing behaviour. This information usually includes the pages a user visits, the time spent on each page, and the objects being interacted with. When the same advertiser appears on multiple websites visited by the user, it can then retrieve the same cookie and update it with new collected information during each visit. Over time, the advertiser can build a profile for the user, with varying levels of detail. This profile can then be leveraged for several purposes, including the provision of personalised ads relevant to the user's interests. This process is commonly known as *profiling*.

In recent years, significant privacy concerns about cookies and profiling have emerged, as the user profiles created by advertisers have reached alarming levels of detail [1, 2, 3]. The main cause is that third parties have full control over the data stored inside the cookies they create, which may include global user identifiers (such as IP addresses) and detailed lists of specific pages visited. Advertisers can then collect this information on a large scale, often without user knowledge or consent [24, 25], in order to not only build detailed user profiles, but also partially reconstruct their browsing history. This history becomes increasingly more accurate as the advertiser is embedded into more websites the user visits. An example of this exploit is shown in Figure 2.1. Once obtained, this browsing history—whether complete or partial— can be used by advertisers to gain more accurate customer insights, thus improving the predictions for future behaviour and increase the effectiveness of targeted advertising. However, it has also the potential to reveal very sensitive details about users, such as their political views, health conditions and personal relationships. Such data can be sold to other companies for monetary gain, or even be exploited by malicious entities for social engineering attacks, through targeted phishing e-mails and other fraudulent communications.

In response to these concerns, regulatory frameworks such as the GDPR [9], the CCPA [10] and the LGPD [11] were introduced in Europe, California and Brazil,

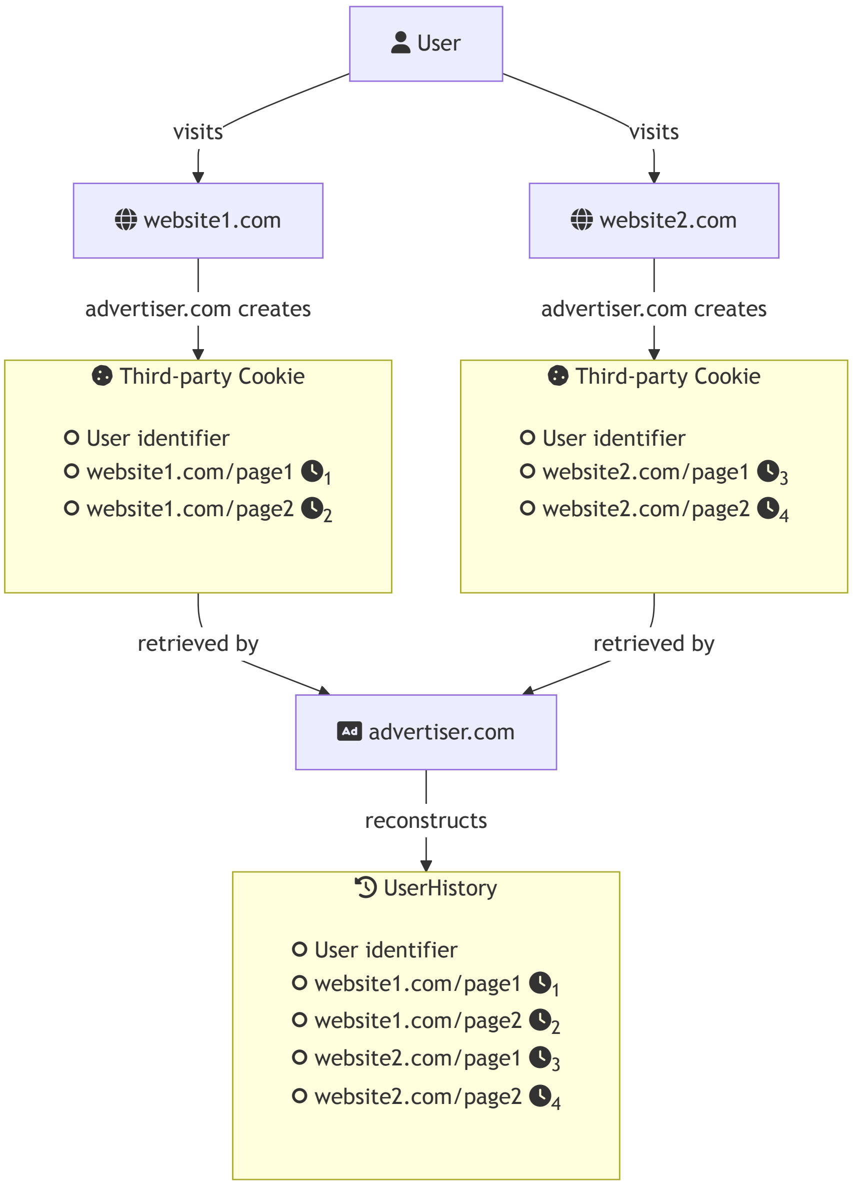


Figure 2.1: Example showing an advertiser using third-party cookies to track a user's activity across two websites they visit. It is implied that both the websites embed the same advertiser inside their page.

respectively. The GDPR and LGPD share similar principles, aimed at giving users the rights to access, correct and delete their personal data while requiring informed consent before third parties can collect this data for profiling. In contrast, the CCPA provides California residents the rights to know about, delete and opt-out of the sale of their data to external companies.

Typically, user consent is gathered by means of *privacy banners* — Web page elements that present users with the option to allow or deny the use of their browsing information through first- and third-party cookies, often accompanied by a link to the website's privacy or cookie policy. However, these banners tend to be invasive, and research has shown that users usually deny consent when given the choice [26], which can be harmful to advertising companies. The introduction of privacy banners has also led to the creation of CMPs (Consent Management Platforms) — services that simplify the creation, customization and embedding of privacy banners inside a first-party website. Well-known examples of CMPs are OneTrust [27], HubSpot [28] and CookieBot [29].

The growing concerns about the potential abuse of third-party cookies encouraged major companies like Mozilla, Apple and Google to shift away from them, and look for alternative solutions for their respective browsers. Each company has taken a different approach:

- Mozilla began blocking third-party cookies by default in Firefox and introduced a new feature called Privacy-Preserving Attribution (PPA), which aims to help advertisers measure ad effectiveness while preserving user privacy, by relying on attribution reports generated directly by the browser [30].

- Apple, like Mozilla, has also implemented the default blocking of third-party cookies in Safari and added a new browser setting called Privacy Preserving Ad Measurement, which behaves similarly to Firefox's PPA [31].

- Google, in contrast, has taken a different approach by building a complete privacy-preserving framework in substitution to third-party cookies, known as the Privacy Sandbox. Its components and functionalities will be detailed in Section 2.2. Google had initially aimed at gradually deprecating their usage through a phase-out process, starting from affecting 1% of Chrome users [14]. However, in July 2024, the company abruptly interrupted the phase-out, instead opting to preserve the usage of third-party cookies and provide users with a more informed choice on their usage [16].

2.2 Google's Privacy Sandbox

The Privacy Sandbox is a family of components designed to create a heathier advertising ecosystem, as a replacement to the current one based on third-party cookies and other profiling techniques, such as browser fingerprinting [32, 33]. It provides advertisers with tools that allow them to display relevant ads based on user interests and measure their effectiveness, while also providing stronger privacy boundaries for information sharing across websites, resulting in greater protection of the users' personal data. Together with the Topics API, these tools include:

- *Attribution Reporting API*: this component allows advertisers to measure the effectiveness of their ads without revealing the individual users' data, by generating aggregated reports directly from the browser. It serves as Chrome's alternative to Firefox's PPA and Safari's Preserving Ad Measurement.

- *Private State Tokens API*: designed to combat spam and fraud on the Internet, this tool allows services to distinguish humans from bots through tokens created without the need of passive tracking, like that being used by CAPTCHA services.

- *Related Website Sets API*: it provides a way for companies to declare relationships between multiple domains, so that browsers can allow the partial sharing of third-party cookies between related sites. This is particularly useful for companies with multiple registered domains (e.g., `advertiser.it` and `advertiser.com`).

- *Shared Storage API*: this component introduces a new form of shared browser storage that can be accessed from multiple websites. To prevent data leakage, the stored data can only be accessed from a secure JavaScript environment, known as a shared storage worklet.

- *CHIPS (Cookies Having Independent Partitioned State) API*: this component partitions third-party cookies based on the website the user is visiting, meaning that a single third party will have separate cookie storage for each website where it is present. This prevents the third party from using the same cookie across different sites, therefore limiting cross-site tracking, while still allowing certain necessary use cases like login status, preferences, or shopping carts to function correctly in scenarios that span multiple domains.

- *Fenced Frames API*: this component introduces a new `<fencedframe>` HTML element, which embeds content similarly to an iframe but prevents the sharing of data between the embedding page (the website) and the embedded content (the advertiser).

- *Federated Credential Management API*: this component allows users to log into sites using federated authentication (e.g., "Sign in with...") without sharing personal information with the identity service or the site. The browser facilitates the user's login from a graphical interface that lets them choose their identity. After that, the browser will perform the login with the Identity Provider without sharing cookies with it.

- *Protected Audience API*: this component enables IBA while protecting the users' privacy. Users are assigned into interest groups, based on the websites they visit and their interactions. Then, when a user visits a page that displays ads, the website (the "seller") runs an auction within the browser to choose the most appropriate ad to display based on the user's interests. Only the winning ads are shown to the user, and the advertiser (the "buyer") can optionally request the browser for a report on the auction's outcome.

- *FLoC (Federated Learning of Cohorts)*—discontinued: similarly to the Protected Audience API, FLoC has the main objective of providing IBA in a privacy-friendly manner. Instead of interest groups, users are automatically placed into cohorts by the browser, based on the websites they visit. The user's cohort is usually shared with the advertiser, so that it can select the most appropriate ad to display. This component, detailed in Section 2.3, is considered as the predecessor of the Topics API.

At the moment of writing, all listed components, except FLoC, have reached the status of general availability, meaning that they are already available to at least 99% of all current Chrome users [21].

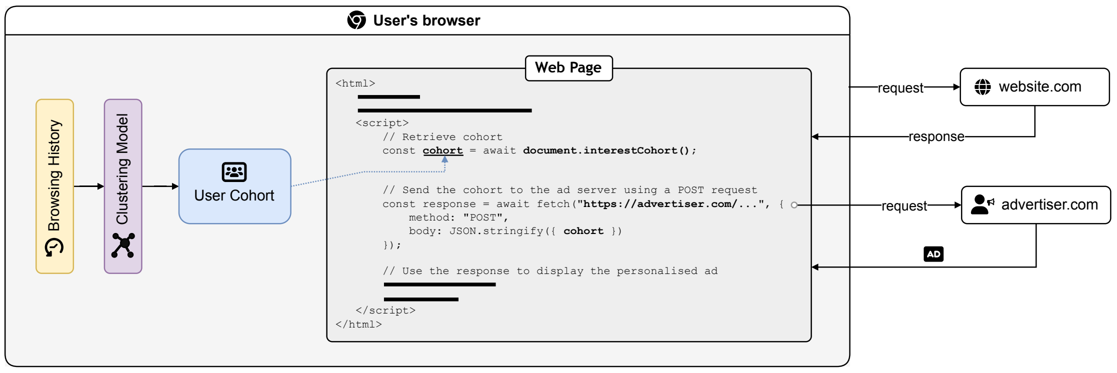
2.3 FLoC

FLoC (Federated Learning of Cohorts) was Google's first attempt at offering a privacy-preserving solution for interest-based advertising. Introduced in March 2021 as a component of the Privacy Sandbox framework (discussed in Section 2.2), it was designed as an alternative to third-party cookies for tracking the users' interests while respecting their privacy.

The functioning principle of FLoC revolves around clustering the users into *cohorts*, based on their browsing history. This is achieved by means of a clustering algorithm developed by Google. Each cohort englobes a large amount of users, making it difficult to identify a single person based solely on the cohort they belong in.

Whenever a user visits a website, a third-party platform can interrogate the browser to retrieve the cohort the user belongs in, by invoking a specific function from a script (namely, `document.interestCohort()`). The user's cohort and the set of possible cohorts are updated weekly. This allows external advertising services to draw conclusions on the user's interests based on their cohort, without receiving any information about the specific websites visited.

FLoC has been heavily criticised throughout its lifetime [17, 18, 19]. While it offered improvements over traditional third-party cookie tracking, there were still concerns about potential user identification across multiple websites. Although



```
<html>

    <script>
        // Retrieve cohort
        const cohort = await document.interestCohort();

        // Send the cohort to the ad server using a POST request
        const response = await fetch("https://advertiser.com/...", {
            method: "POST",
            body: JSON.stringify({ cohort })
        });

        // Use the response to display the personalised ad

    </script>
</html>
```

Figure 2.2: Simplified schema of FLoC's mechanisms.

a single cohort is linked to many users, the set of cohorts the user belongs to becomes increasingly unique as time passes. Therefore, a malicious third party could collect a user's cohorts over an extended period of time across multiple websites and cross-reference the collected lists: the longer the overlap, the greater the probability that the same user has visited the those websites. For this reason, Google discontinued the development of FLoC starting from January 2022, in favour of the Topics API. Chapter 3 provides an in-depth view on how the new successor works.

Chapter 3

The Topics API

The Topics API [34, 35] is the evolution of the FLoC inside the Privacy Sandbox. Similarly to its predecessor, this component was introduced to provide a privacy-friendly solution for interest-based advertising. Compared to FLoC, the Topics API replaces user cohorts with pre-defined *topics* of interest. A user can have multiple topics assigned, based on the websites they visit during a given amount of time. To preserve their privacy, both the amount of topics assigned to users and the ones disclosed to an advertising platform is limited.

At the time of writing, the Topics API is available on two main platforms: the Web and Android. The Web version is intended to support interest-based advertising within Web applications, preventing the usage of third-party cookies for cross-site tracking. It is currently only available on the mobile and desktop versions of Chromium and Chrome browsers, starting from version 101 released in March 2022. On the other hand, the Android version provides the same features of its Web counterpart, but is designed for mobile applications running on the Android operating system, preventing advertisers from tracking users across multiple applications. This chapter will focus on the Web version, detailing its mechanisms in the following sections.

3.1 Terminology

This section will introduce some key terms needed to fully comprehend how the Topics API works.

- A *topic* represents the classification of a website with a user's topic of interest.
- A *taxonomy* is a comprehensive list of topics that follows a hierarchy, grouping multiple topics into categories. For example, a topic of interest defined in the taxonomy is `News`, which refers to news platforms in general. This topic also

represents a category, optionally allowing more detailed classifications such as `News/Local News` and `News/World News` to identify the specific types of news. At the moment of writing, the most recent taxonomy version used by Google is 2[1].

- A *caller* or CP (Calling Party) refers to a third-party service that performs a call to the Topics API. This role is usually taken by advertisers that are embedded in a website.

- An *epoch* refers to a certain time interval, which defaults to one week but can be customised by an advanced user by setting a configuration flag inside the browser. The initial time of each epoch is randomised separately for each caller and website.

- An *override list* refers to a data structure that maps pre-defined domains with their "true" topics. At the moment of writing, the topics are mapped for the top 10,000 websites by popularity according to Google.

- A *classifier model* is a machine learning model that associates a user-visited domain to zero or more topics. At the time of writing, the model used by Google for this task is built upon the override list and is based on BERT, a natural language processing model developed by them [36]. Currently, the only input of the model is the website's domain name, leading to possible inaccuracies in the assigned topics, especially if the domain name is ambiguous or misleading [37].

3.2 The algorithm

This section will provide an overview of how the Topics API collects the users' topics of interest and how they are disclosed to the advertising platforms.

First, the browser passively monitors the user's browsing activity, saved locally within Chrome's user profile folder[2] as a local database named `History`. Each website is then assigned a set of topics: if the website appears in the override list, its corresponding topics are chosen; otherwise, the browser uses Google's BERT classifier model [38] to estimate the domain's topics.

[1]The complete taxonomy can be found at `https://github.com/patcg-individual-drafts/topics/blob/main/taxonomy_v2.md`

[2]This folder is typically located at `%localappdata%\Google\Chrome\UserData\Default` on Windows, `/Users/<username>/Library/ApplicationSupport/Google/Chrome/Default` on Mac and `~/.config/google-chrome/Default` on Linux.

At the end of each epoch, the browser computes the top-5 most visited topics by the user and stores them inside a local list, detailed in Section 3.3.1. For each topic, a collection of *observers* is saved, i.e., the third parties that performed a Topics API call on a website whose domain is classified with that specific topic. This information can be easily viewed from the `chrome://topics-internals` page.

When a user visits a website, an authorised caller (details in Section 3.4) can call the Topics API in one of three ways, listed in Section 3.5. The browser will then randomly select a topic for each of the previous three epochs at most, and then disclose the selections to the caller. It's important to note that, during a given epoch, a caller can only learn the topics that have been observed by it in any of the previous three epochs. Therefore, an advertiser that calls the Topics API for the first time inside a browser will not receive any topics, as none were observed by it in the previous epochs. Figure 3.1 illustrates the overall process of topic derivation and disclosure.

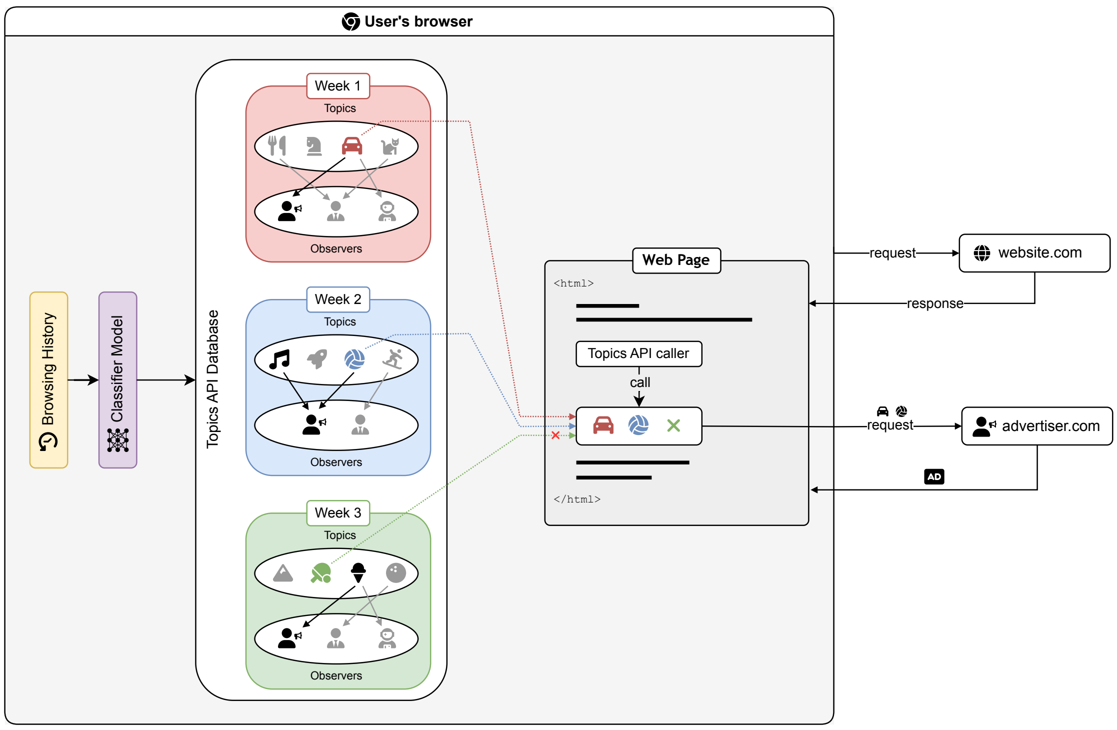


Figure 3.1: Simplified diagram of the Topics API's mechanisms. In the above example, out of the three topics chosen by the browser, only two are disclosed to the advertiser, because the topic drawn in week 3 was not previously observed by them.

To further protect the users' privacy, an element of *plausible deniability* is introduced: there is a 5% chance that the topic will be randomly selected among the entire taxonomy instead of the top-5 list. In that case, the selected topic will be disclosed regardless of whether it had been previously observed by the caller. This makes it more challenging for a malicious entity to build a user's profile and identify them across multiple websites. However, it has been shown that it remains feasible to perform cross-site user identification, by leveraging topic information collected over a long period of time [20, 37, 39, 40]. The process is very similar to what has already been discussed in Section 2.3: a malicious entity could collect the user's topics over several epochs and across multiple websites, then cross-reference these lists to identify the same user across different sites. As more topics are collected, the probability of correctly identifying the user increases. Figure 3.2 illustrates the threat model of user re-identitification through the Topics API. To circumvent the plausible deniability aspect, the attacker could apply a filter that eliminates the topics that appear less frequently inside the collection, as they may potentially refer to randomly selected topics [20].

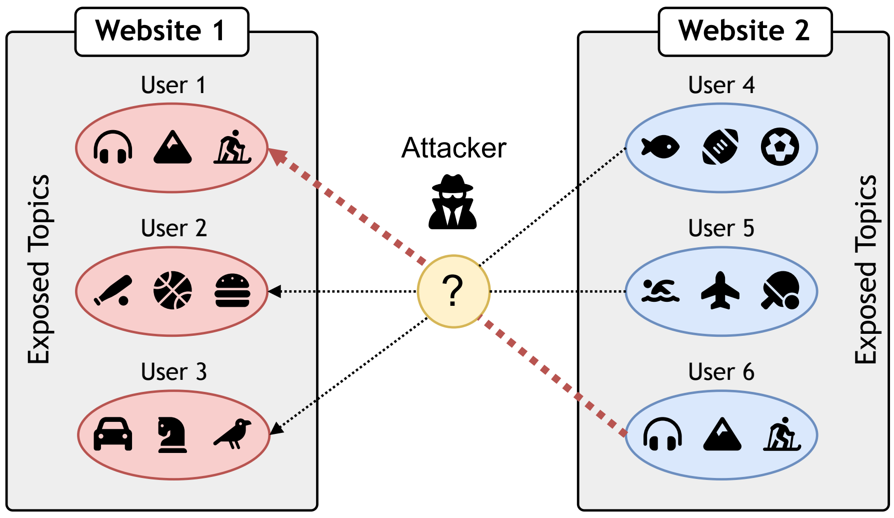


Figure 3.2: Simplified schema showing the cross-site user re-identification threat model within the Topics API.

3.3 Internal data structures

This section will detail the data structures used by the Topics API to store persistent information locally within the user's machine.

3.3.1 BrowsingTopicsState

This JSON file, located inside Chrome's user profile folder, stores the top-5 topics for each epoch. As shown in Listing 3.1, the file consists of the following information:

- The time at which the top-5 list was calculated.
- Metadata regarding the version of the classifier model, the browser's configuration and the taxonomy.
- The list of top-5 observed topics, each containing the list of observers, in hashed form.
- The index of the list from which topics have been generated at random. This happens whenever the user has visited less than 5 topics during the epoch.

```
{
  "epochs": [
    {
      "calculation_time": "123456",
      "config_version": 2,
      "model_version": "5",
      "padded_top_topics_start_index": 2,
      "taxonomy_version": 2,
      "top_topics_and_observing_domains": [
        {
          "hashed_domains": [
            123456,
            ...
          ],
          "topic": 123
        },
        ...
      ]
    }
  ]
}
```

Listing 3.1: Example showing the structure of the `BrowsingTopicsState` file.

3.3.2 BrowsingTopicsSiteData

This file-based relational database, located in Chrome's user profile folder, keeps track of the websites that invoke the Topics API. Its entity-relationship diagram is shown in Figure 3.3. Specifically, this database keeps track of:

- The *context domain*, i.e., the 2LD (Second-Level Domain) of the browsing context that invokes the Topics API. A *browsing context* [41] refers to the environment where a browser displays a document. In modern browsers, this is typically a tab, a window or an iframe.
- The *main frame host*, i.e., the website where the Topics API call took place.
- The timestamp of the *latest usage*, separately by context domain and main frame host.
- Various meta information, such as the version of the database.

An important aspect of this database is that it only keeps track of *successful* Topics API calls. If a call fails or is blocked for any reason, it will not be recorded.

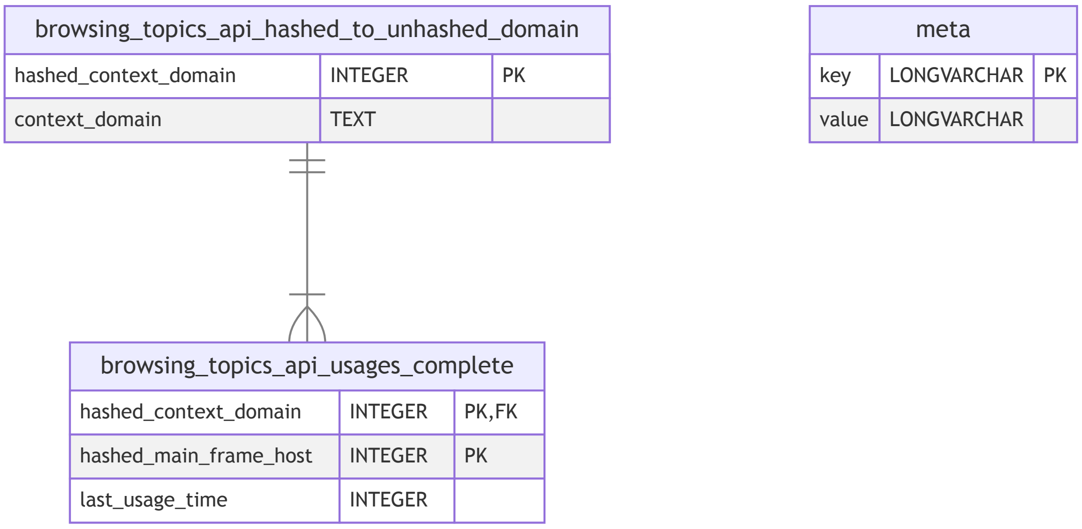


Figure 3.3: Entity-relationship model of the `BrowsingTopicsSiteData` database.

3.3.3 PrivacySandboxAttestations

This data structure is used by the browser to keep track of the third parties that are allowed to call each component of the Privacy Sandbox. It's contained inside a configuration file named `privacy-sandbox-attestations.dat`, located inside the `PrivacySandboxAttestationsPreloaded` folder of Chrome's local configuration[3].

[3]This folder is typically located at `%localappdata%\Google\Chrome\UserData\Default` on Windows, `/Users/<username>/Library/ApplicationSupport/Google/Chrome` on Mac and `~/.config/google-chrome` on Linux.

The contents of this file are encoded as a protocol buffer, a binary format designed by Google [42]. Listing 3.2 shows an example of its contents, converted to a human-readable format.

```
all_apis: ATTRIBUTION_REPORTING
all_apis: PRIVATE_AGGREGATION
all_apis: PROTECTED_AUDIENCE
all_apis: SHARED_STORAGE
all_apis: TOPICS
sites_attested_for_all_apis: "https://360yield.com"
[...]
sites_attested_for_all_apis: "https://yelp.com"
site_attestations {
  key: "https://a-mo.net"
  value {
    attested_apis: PRIVATE_AGGREGATION
    attested_apis: PROTECTED_AUDIENCE
    attested_apis: SHARED_STORAGE
    attested_apis: TOPICS
  }
}
[...]
```

Listing 3.2: Example of `privacy-sandbox-attestations.dat`'s contents, shown in a human-readable format.

3.4 Authorised callers and attestations

Since the Privacy Sandbox handles sensitive user information and could be exploited to potentially breach user privacy, all of its components, Topics API included, are restricted to select third-party platforms that have completed an enrolment process [43]. Upon its completion, each platform receives an *attestation file*—a JSON file containing details such as which Privacy Sandbox components the third party is allowed to use, on which platform and for how long. The structure of the JSON, exemplified in Listing 3.3, is organized in one or more `privacy_sandbox_api_attestations`, each containing the following information:

- The `attestation_parser_version`, which helps determine the properties that are expected within this object. At the moment of writing, the most recent version is 2, featuring minor updates from the first version.
- An `attestation_version`, usually a number that increments with each element in the `privacy_sandbox_api_attestations` collection.

```
{
  "privacy_sandbox_attestations": [
    {
      "issued_time_since_epoch": 123456,
      "expired_time_since_epoch": null,
      ...,
      "platform_attestations": [
        {
          "platform": "chrome",
          "attestations": {
            "topics_api": {
              "ServiceNotUsedForIdentifyingUserAcrossSites": true
            },
            ...
          }
        },
        ...
      ]
    }
  ]
}
```

Listing 3.3: Structure of a valid attestation JSON file.

- A `privacy_policy`, specified as a list of URLs to textual documents.
- An `enrollment_id` and `ownership_token`, used by Google for verification purposes.
- The `enrollment_site`, which specifies the domain of the third party allowed to call the Privacy Sandbox components. Interestingly, we have noticed that the enrolment site does not always coincide with the domain where the attestation file is placed.
- The time window of validity, defined by the fields `issued_time_since_epoch` and `expiry_time_since_epoch`. The latter field is optional.
- A collection of `platform_attestations`, JSON objects detailing for each platform (e.g. Chrome, Android) a list of the Privacy Sandbox components that the third party is allowed to invoke. For each component, the third-party platform states through an *attestation* in the form of a boolean property (namely, `ServiceNotUsedForIdentifyingUserAcrossSites`) that it will not exploit the component for cross-site user identification.

This file must be made publicly accessible at a specific path within the platform's domain, namely `<domain>/.well-known/privacy-sandbox-attestations.json`.

In theory, failure to do so should result in the revocation of permission to use any Privacy Sandbox component. Overall, this mechanism provides a way to easily determine who can access the Topics API and other components of the Privacy Sandbox. This information is not only available to Google for access control purposes, but also to users for transparency.

The browser maintains a local record of every authorised third party, along with the specific Privacy Sandbox components each is allowed to use, inside the `privacy-sandbox-attestations.dat` data structure, described in Section 3.3.3. When any component of the Privacy Sandbox is called, the browser compares the caller's domain against the list: the request is processed only if the domain appears in the list and is allowed to call the specific API. The sandbox attestation list is automatically kept up to date by the browser, as its contents are updated daily by Google. In theory, only platforms that have completed the enrolment process and correctly exposed the JSON attestation file within their domain are granted inclusion in this list. Conversely, failing to do so, even after being included, should result in their removal from the list.

3.5 Usage types

The Topics API can be called from the browser in any of three ways. Each of them will be listed and detailed in this section.

3.5.1 JavaScript

The Topics API can be called by means of an asynchronous JavaScript function (namely, `document.brosingTopics()`). This function resolves with an array of up to 3 objects, representing the user's topics of interest. This result can then be shared with the third party's server through an HTTP exchange. The following code snippet shows a simplified example of implementation:

```javascript
// Retrieve the topics
const topicsArray = await document.browsingTopics();

// Send the topics to the Ad server
const response = await fetch("https://advertiser.com/get-ad", {
    method: "POST",
    body: JSON.stringify({ topics: topicsArray })
});

// Use the response to display the Ad...
```

Each object in the `topicsArray` contains:

- The `topic`, represented as an integer identifying the topic in the taxonomy.
- The `version`, which is usually expressed as three separate properties (namely `configVersion`, `modelVersion` and `taxonomyVersion`), but this information can also be contained in a unique property concatenating the three values.

3.5.2 Fetch

The Topics API can be called directly through JavaScript's `fetch` function, by setting the dedicated `browsingTopics` boolean option. The previous example code can be re-written as the following:

```
// Send the topics to the Ad server
const response = await fetch("https://advertiser.com/get-ad", {
    method: "GET",
    browsingTopics: true
});

// Use the response to display the Ad
```

Behind the scenes, the browser sets a dedicated HTTP header in the request, containing the user's topics of interest and version information. An example value of this header, containing topics 123 and 456, is the following:

`Sec-Browsing-Topics:(123 456);v=chrome.2:2:2, ();p=P000...`

The `p` value in the header serves as a padding, to ensure that the header's length remains consistent. The server can then read the contents of this header and return an appropriate ad to show the user. In order to have the browser mark the topics as observed by the third party, the server must contain the following HTTP header in its response:

`Observe-Browsing-Topics:?1`

3.5.3 IFrame

The Topics API can be called directly from the HTML of the website's page, by integrating an `iframe` tag with the `browsingtopics` attribute set. Following the previous example, the JavaScript code can be omitted in favour of the following `iframe` tag inside the Web page's HTML:

```
<iframe src="https://advertiser.com/get-ad" browsingtopics></
   iframe>
```

Behind the scenes, the same header introduced in Section 3.5.2 is set inside the request to the server. Therefore, this method allows to perform the same operations as in Section 3.5.2 while writing less code, if the advertiser returns an HTML page for display.

Chapter 4

Methodology for data collection

This chapter will describe in detail the methods adopted to collect measurements related to the Topics API across the Web. The data collection process is performed in form of a measurement campaign, divided into multiple steps. Each step, summarised in Figure 4.1, will be briefly explained below and further detailed in the following sections.

1. *Website crawling with Priv-Accept*: each website is crawled using *Priv-Accept*, a Selenium-based Web crawler created for a previous study [23]. This tool was modified to allow the collection of data related to the Topics API. For each website visited, the crawler outputs a JSON file containing useful information collected during the crawling, such as the third parties encountered and the Topics API calls recorded. Section 4.1 provides details on how the crawler works and how it was adapted for this campaign.

2. *Extract contacted domains*: this stage extracts, starting from the previous stage's outputs, the entire list of websites and third parties contacted during the crawling process. The reasons for this step are detailed in Section 4.3.

3. *Extract* Attested *domains*: this stage filters the *Attested* domains from the list obtained in the previous step. This is done by verifying the presence of the relative attestation file at the `./well-known/...` path, mentioned in Section 3.4. More information about how a domain is classified as *Attested* can be found in Section 4.3.

4. *Extract* Allowed *domains*: the set of *Allowed* domains is extracted from the `privacy-sandbox-attestations.dat` list using a protocol buffer library. Further details can be found in Section 4.3.

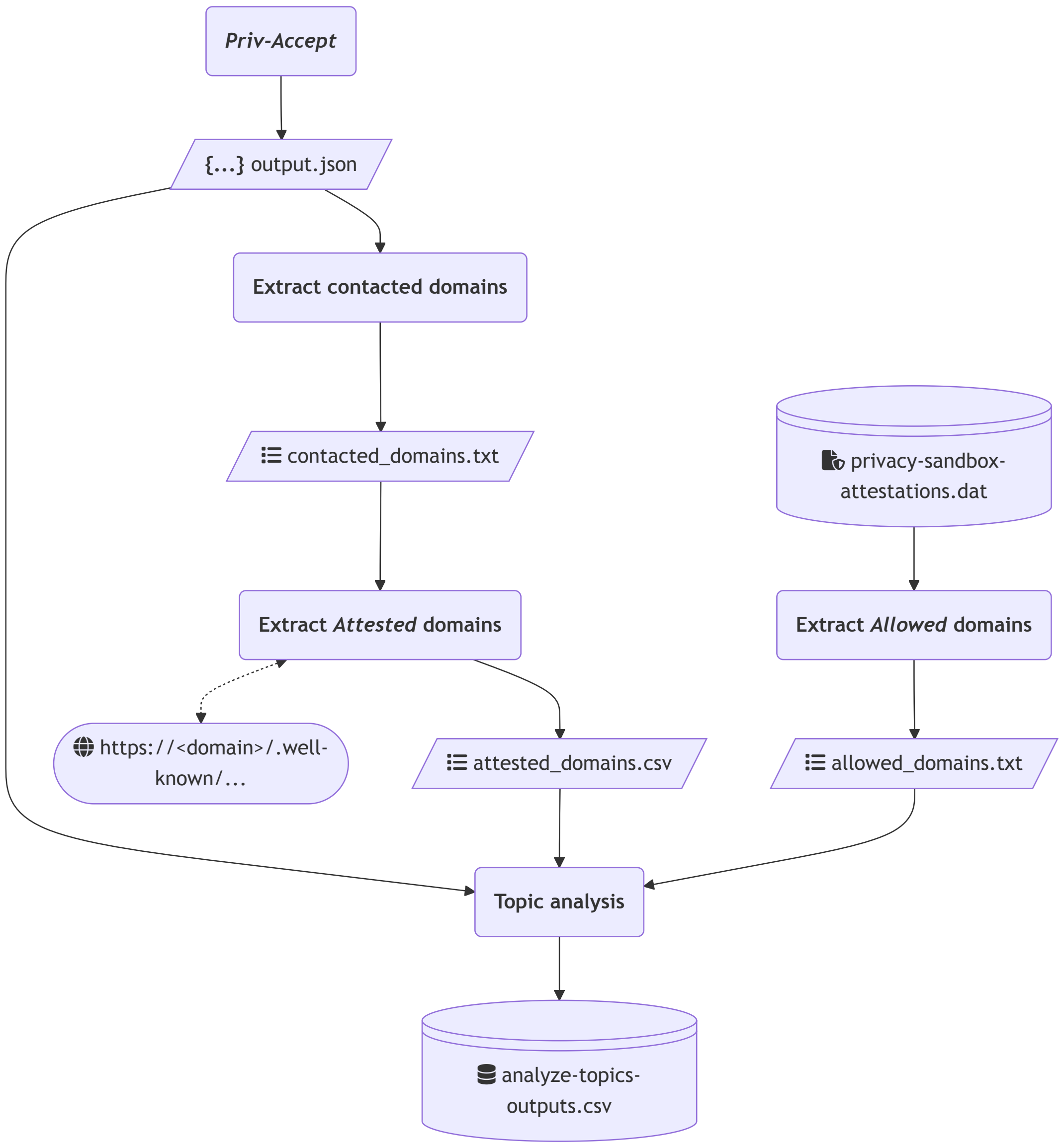


Figure 4.1: Overview of the data collection process.

5. *Topic analysis*: this final stage filters the most relevant information for the study and saves it into a more compact JSON file for each website crawled. Specifically, it stores the list of Topics API usages detected, the list of *Attested* and *Allowed* domains encountered, and whether a privacy banner was found and clicked by the crawler. The various final outputs can then be further compacted into a single CSV file. More details on this step can be found in Section 4.4.

4.1 The *Priv-accept* Web crawler

Priv-Accept is a Selenium-based Web crawler. It was developed for a previous study that analysed the behaviour of Web pages before and after a user has provided consent for data collection [23]. The crawler has the unique feature of searching for privacy banners on websites and clicking the "Accept" button, thus mimicking the behaviour of a user accepting a website's privacy policy. This step is of utmost importance when studying the Topics API in real-world applications, especially because the measurements are performed from Europe, where the GDPR is in effect. Given the role of the Topics API in handling sensitive user information, we expect that the technology must follow the same regulatory frameworks for privacy protection as third-party cookies. In practice, this means that users must agree to a website's privacy policy and explicitly authorise the use of their personal data before any third party can collect information after a Topics API call. Moreover, [23] shows that a significant amount of trackers and other third-party services are contacted only after accepting the visited website's privacy policy, if present. Therefore, following this approach would lead to more accurate results, and would also allow to compare the third parties' behaviour before and after accepting a website's privacy policy.

The original version of *Priv-Accept* used in [23] performs the crawling in the following steps:

1. *Optional warm-up visit*: the crawler loads the website in advance, to populate the browser cache. While useful for [23], it's not essential in this case, as the cache will be cleared before every visit.

2. *First visit*: the crawler loads the website for the first time, collecting statistics such as the contacted third parties. We call this step *Before-Accept*, as in [23].

3. *Search privacy banner and click the "Accept" button*: the crawler searches for a privacy banner, by scanning the page for "Accept"-like words. These words reflect the possible labels of a consent banner's "Accept" button and are defined in a customisable list. If an "Accept" button is found, the crawler attempts to click it.

4. *Second visit*: after clicking the "Accept" button, the crawler refreshes the page and collects the same set of statistics as in *Before-Accept*. We call this step *After-Accept*, as in [23].

5. *Optional additional visits*: the crawler visits a custom number of internal pages, randomly chosen from the internal links[1] found within the visited page. The same statistics as in *Before-Accept* and *After-Accept* are collected for each internal page visit. This step is not very significant for this study, as the number of third parties does not typically increase after visiting the internal pages.

Priv-Accept was modified in order to allow the recording of Topics API calls. To permit the reproducibility of our results, we leave the edited source code available to the public.[2]

The method of collection takes advantage of the `BrowsingTopicsSiteData` database described in Section 3.3.2: its contents are retrieved using a SQLite library [44] and then converted into JSON format, which can be saved into the crawler's final output. To expand upon the limited information originally collected from this database, we modified Chromium's source code and built a custom version that stores additional information. The specific details of the modifications will be provided in Section 4.2. Using this customised version of Chromium, we are now able to collect the following information from the local database for each Topics API invocation:

- The *context origin*, i.e., the FQDN of the browsing context where the call takes place.
- The *caller source* or usage type, as outlined in Section 3.5.
- The *timestamp* of the call.

The crawler outputs one JSON file per visited website, containing detailed information about each stage of the crawling:[3]

- The collection of URLs that have been contacted by the browser during the visit, including the website itself and any third parties.
- The list of cookies set during the visit, along with relevant information such as their value, expiration, and source. This data is sufficient to determine, for instance, the type of cookie (i.e., persistent or session, first- or third-party).
- The list of Topics API usages detected.

[1]Internal links refer to links that have the same FQDN (Fully Qualified Domain Name) as the visited website.

[2]Available at `https://github.com/Novant8/priv-accept-topics`.

[3]The stages are, in order: `pre-visit`, `first`, `click`, `second`, `internal`. The first and the last of them are optional.

Additionally, the output contains a list of DOM elements that have been found to be candidates for a possible "Accept" button, and which of them—if any—has been clicked. This information will be used to determine whether a privacy banner was found. Finally, the output file contains page loading statistics and the crawler's logs, for troubleshooting purposes.

It must be noted that *Priv-Accept* has some limitations. Primarily, its privacy banner detection is not entirely accurate, as our current "Accept" keyword list supports only five languages: English, French, Spanish, German and Italian. Since we are considering a list of *global* websites, there is a high chance that websites in other languages will appear. In such cases, *Priv-Accept* may fail to recognise the "Accept" button even if a privacy banner is present, and the result will appear indistinguishable from that of a website without one. Moreover, the measurement campaign is conducted exclusively from Europe, which could result in websites being blocked by local security mechanisms such as firewalls and other domain-blocking services running on the Interner Service Provider's (ISP) part. For that reason, the actual number of websites that produce results will probably be lower than 50,000. A potential topic of future research could involve the conduction of the same campaign from different locations, to see how the results compare.

4.2 Modifications to the Chromium browser

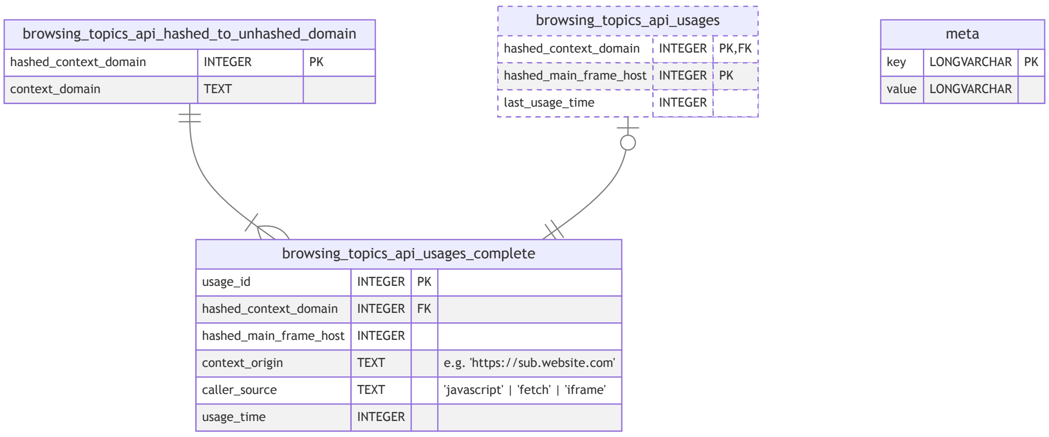
As introduced in Section 4.1, the crawler uses a customised version of Chromium that obtains additional Topics API usage information that are useful for this study. This section details the specific modifications performed to the C++ source code of Chromium, version `122.0.6261.128` [45].

The primary goal of these modifications is to expand the already existing `BrowsingTopicsSiteData` database by adding fields that may be useful for this and other relevant studies. The new fields include:

- The *context origin*, i.e., the FQDN of the third party that performed the call.
- The *caller source*, i.e., the usage type as introduced in Section 3.5.

In addition to these new fields, the modifications allow to save the information related to *every* call, instead of just the most recent one for a given party, as is currently the case.

To achieve this, it is necessary to modify the existing database's structure: the target structure's entity-relationship model is shown in Figure 4.2. To reach this schema, the context origin and caller source can simply be added inside the `browsing_topics_api_usages` table as new attributes. However, to allow the collection and saving of every call, a unique `usage_id` attribute must be introduced as the new primary key. To prevent breaking other browser components that may depend on the old database structure, all new and existing attributes are saved inside a physical table called `browsing_topics_api_usages_complete`. The original `browsing_topics_api_usages` table is then redefined as a virtual view of the complete table.



browsing_topics_api_hashed_to_unhashed_domain		
hashed_context_domain	INTEGER	PK
context_domain	TEXT	

browsing_topics_api_usages		
hashed_context_domain	INTEGER	PK,FK
hashed_main_frame_host	INTEGER	PK
last_usage_time	INTEGER	

meta		
key	LONGVARCHAR	PK
value	LONGVARCHAR	

browsing_topics_api_usages_complete			
usage_id	INTEGER	PK	
hashed_context_domain	INTEGER	FK	
hashed_main_frame_host	INTEGER		
context_origin	TEXT		e.g. 'https://sub.website.com'
caller_source	TEXT		'javascript' | 'fetch' | 'iframe'
usage_time	INTEGER		

Figure 4.2: Updated entity-relationship diagram of the `BrowsingTopicsSiteData` database. The `browsing_topics_api_usages` entity represents a virtual view over the `browsing_topics_api_complete` table.

The database is managed by the `BrowsingTopicsSiteDataStorage` class.[4] This class contains two key functions for the database's management:

- Function `CreateSchema`[5] is invoked whenever the browser is opened and the database is either corrupted or not available. Its purpose is to initialize the database, by defining its structure. In fact, it contains the Data Definition Language (DDL) that creates the needed tables and does not take any parameters.

- Function `OnBrowsingTopicsApiUsed`[6] is invoked on every Topics API call by

[4]The implementation of this class is available at `https://source.chromium.org/chromium/chromium/src/+/refs/tags/122.0.6261.128:content/browser/browsing_topics/browsing_topics_site_data_storage.cc`

[5]The source code of this function is available at `https://source.chromium.org/chromium/chromium/src/+/refs/tags/122.0.6261.128:content/browser/browsing_topics/browsing_topics_site_data_storage.cc;l=309`

[6]The source code of this function is available at `https://source.chromium.org/chromium/chromium/src/+/refs/tags/122.0.6261.128:content/browser/browsing_topics/browsing_topics_site_data_storage.cc;l=169`

an authorised party. It inserts or updates a usage entry in the database and takes three parameters: the digest of the main frame host, the context domain and its hashed form.

Both functions were modified to reflect the new database structure. While modifying `CreateSchema` was relatively simple, `OnBrowsingTopicsApiUsed` was more complex, due to its parameters not supporting the retrieval of the required information. For this reason, we adapted the code to leave the task of collecting the context origin and caller source to the other functions that invoke `OnBrowsingTopicsApiUsed`, shown in Figure 4.3. The retrieved fields are then passed down to `OnBrowsingTopicsApiUsed` as extra function parameters.

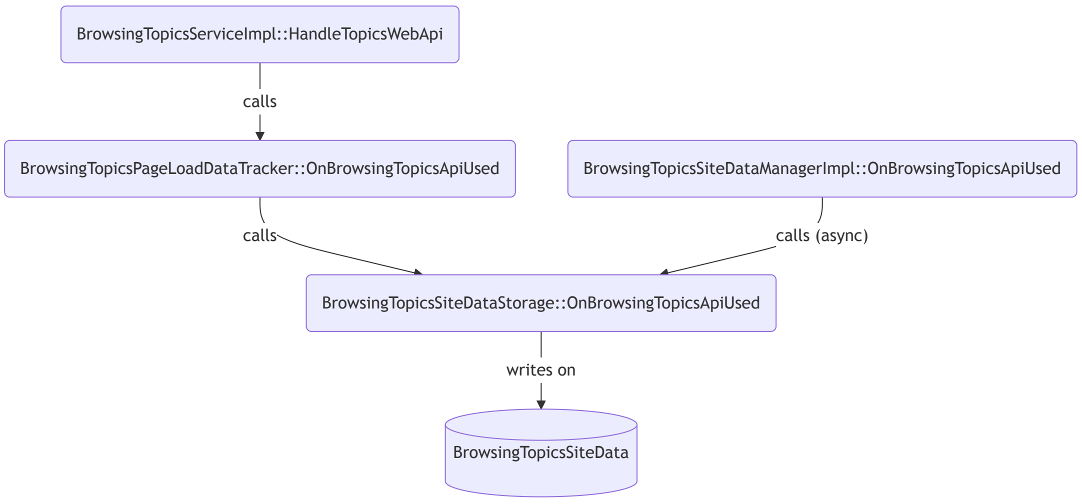

BrowsingTopicsServiceImpl::HandleTopicsWebApi
calls
BrowsingTopicsPageLoadDataTracker::OnBrowsingTopicsApiUsed
BrowsingTopicsSiteDataManagerImpl::OnBrowsingTopicsApiUsed
calls
calls (async)
BrowsingTopicsSiteDataStorage::OnBrowsingTopicsApiUsed
writes on
BrowsingTopicsSiteData

Figure 4.3: Dependency graph of functions that call `OnBrowsingTopicsApiUsed` and were modified for the purpose of this analysis.

The full code snippets of the most relevant functions modified are available in Appendix A. Moreover, the full changes to Chromium's code, with respect to version `122.0.6261.128`, are left available to the public.[7]

After the applied modifications, we built the new version of Chromium by following their guide [46].

[7]The changes are available at `https://github.com/Novant8/priv-accept-topics/blob/main/chromium-changes.patch`. They can be applied to Chromium's source code by running Linux's `patch` command at its root folder.

4.3 Domain extraction and attestation

When a Web browser visits a website, numerous files are downloaded from multiple sources, many of which represent third-party services such as online advertisers. One of the key aspects of this study is determining which of these third parties are authorised to invoke the Topics API. As outlined in Section 3.4, there are two main ways for demonstrating this:

- Verifying whether the 2LD of the third party contains a valid attestation file at the `.well-known/...` path.

- Checking whether the 2LD of the third party is present inside the local `privacy-sandbox-attestations.dat` list downloaded by the browser.

We will use both methods, to detect possible inconsistencies. For clarity, we refer to domains meeting the first condition as *Attested* and those meeting the second condition as *Allowed*.

To determine the set of *Attested* websites, all domains encountered during the measurement campaign need to be contacted. To avoid contacting the same domain multiple times, this process is performed on the set of *unique* domains retrieved from all of *Priv-Accept*'s outputs. This set is saved into a text file called `contacted_domains.txt`. For each domain in this list, a Python script sends an HTTP request to the URL `https://<domain>/.well-known/privacy-sandbox-attestations.json`. If the server responds with some content, the script verifies whether it is a JSON file with the same structure as described in Section 3.4. Specifically, for a domain to be considered *Attested*, the attestation JSON must contain a non-expired Privacy Sandbox attestation with an attestation object for Chrome's platform for the Topics API. In other terms, all the properties shown in Listing 3.3 must be present, with `expired_time_since_epoch`'s value being `null` or greater than the current time. The set of *Attested* domains is saved into a separate CSV file called `attested_domains.csv` which contains, for each row, the *Attested* domain and the full content of the JSON attestation file.

To determine the set of *Allowed* websites, the process is more straightforward: the list can be retrieved directly from the `privacy-sandbox-attestations.dat` file, introduced in Section 3.3.3, using a protocol buffer library [42]. Since not all the domains listed in the file are authorised to use the Topics API, it is necessary to filter out those that lack the permissions for the component. This step outputs a text file called `allowed_domains.txt`, containing one *Allowed* domain per row.

4.4 Topic analysis

Priv-Accept's outputs can become particularly large, especially when full request and response logging is enabled. Since only a limited amount of the information stored in these outputs is actually needed for this study, this final step of the anaysis is introduced to filter out the unnecessary data, producing a more compact final output. This is done by means of a Python script, whose behaviour can be summarised in the following three operations:

- *Extract* Allowed *and* Attested *domains*: for each website and visit, we extract the list of *Attested* and *Allowed* websites. This is done by cross-referencing the domains contacted during the visit with the overall list of *Attested* and *Allowed* domains, obtained as explained in Section 4.3.
- *Extract banner data*: this step obtains directly from the output of *Priv-Accept* whether the privacy banner was found and clicked, for each website visited.
- *Extract Topics API usages*: for each website and visit, the list of Topics API usages is collected directly from the *Priv-Accept* output.

The structure of the final output for each visited website is summarised in Listing 4.1. This format allows the file to be easily converted into the row of a CSV file, for more efficient storage.

```
{
  "url": "https://website.com",
  "banner_clicked": true,
  "first": {
    "attested_domains": [
      "advertiser.com",
      ...
    ],
    "allowed_domains": [
      "advertiser.com",
      ...
    ],
    "topics_api_usages": [
      {
        "context_origin_url": "https://sub.advertiser.com",
        "caller_source": "javascript",
        "usage_time": 123456,
        "possible_callers": [
          {
            "url": "https://advertiser.com/script.js",
            "reason": "browsing-topics-in-script"
          },
          ...
        ]
      },
      ...
    ]
  },
  "second": {
    ...
  }
}
```

Listing 4.1: Example showing the final output's JSON structure relative to a single visited website. The `first` property refers to the *Before-Accept* visit, while `second` refers to the *After-Accept* visit.

Chapter 5

Dataset and results

We conduct the first iteration of the measurement campaign on March 30^{th}, 2024, from a single machine part of the Politecnico di Torino's computing cluster. The crawler visited the top 50,000 websites by popularity according to the Tranco list, as of March 26^{th}, 2024 [47]. The campaign lasted for about 36 hours.

At the end of this first iteration, we identify two main datasets:

- Out of the total 50,000 attempted visits, 43,405 successfully produce an output for at least the *Before-Accept* visit. The remaining websites fail mainly due to domain name resolution and connection-related errors. It also includes 19,534 third parties encountered during the crawling. We refer to this dataset as D_{BA}.
- Of the websites in D_{BA}, only 14,719 (about 30%, which appears to be in line with [23]) contained a privacy banner that *Priv-Accept* was able to find and provide consent to the usage of personal information. As mentioned in Section 4.1, this does not reflect the *true* number of websites with a privacy banner, as the crawler could have missed the keyword or visited a website in an unsupported language. We refer to this dataset as D_{AA}.

With regard to the Topics API calls, we find a total of 1,337 callers during *Before-Accept*, and 2,662 during *After-Accept*. From this point, they will be referred to as CPs (Calling Parties). Table 5.1 summarises the most interesting numbers obtained:

- 193 domains appear to be *Allowed*. In theory, these should be the only ones able to perform a Topics API call.
- Out of the 193 *Allowed* domains, 181 expose the attestation file at the correct path, while 12, erroneously, do not. We analyse the enrolment timeline of the 181 *Attested* and *Allowed* third parties in Section 5.1.

	CP	***Allowed***	***Attested***	
	-	✓	-	193
	-	✓	✗	**12**
D_{AA}	✓	✓	✓	47
D_{AA}	✗	✓	✓	105
D_{AA}	✓	✗	✓	**1**
D_{AA}	✓	✗	✗	**2,614**
D_{BA}	✓	✓	✓	**28**
D_{BA}	✓	✗	✓	**1**
D_{BA}	✓	✗	✗	**1,308**

Table 5.1: Overall status of the Topics API callers encountered during the measurement campaign. Each row counts the number of third parties encountered, taking into account whether they are CPs, and whether they are inside the *Allowed* and *Attested* sets. The first collection of rows refers to the information obtained from the local browser allow-list (Section 3.3.3) and the JSON attestation files (Section 3.4). The other two groups refer to the third parties present inside the D_{AA} and D_{BA} datasets, respectively. We highlight, in red, the anomalous usage and, in blue, the questionable usage.

- During the crawling process, we encounter a total of 152 allowed third parties in D_{AA}, out of which only 47 call the Topics API. We detail the *legitimate* usages performed by these CPs in Section 5.2.

- Surprisingly, we find a very large number—thousands—of websites and CPs that call the Topics API, even if they are not part of the *Allowed* or the *Attested* set. These *anomalous* usages and the possible reasons of their existence will be discussed in Section 5.3.

- We find that one CP—namely `dstillery.com`—contains a valid attestation file, issued in November 2023, but is not contained in the *Allowed* set. This possibly reflects that the attestation process is still ongoing, or that Dstillery is not interested in completing it. At the moment of writing, the status of this CP remains unchanged.

- In D_{BA}, we would expect to encounter no call to the Topics API, given that the user hasn't accepted the privacy policy yet. However, we find a total of 1,337 parties that performed a call before the privacy policy was accepted. Out of these, 28 are from CPs that are *Allowed*, 1,308 are from CPs that are not *Allowed* or *Attested*, and one—namely, `tail.digital`—is from a CP that is *Attested* but not *Allowed*.[1] We discuss the possible reasons of these *questionable* usages in Section 5.4.

The following sections will provide a global picture of the Topics API's presence in the current Web ecosystem, providing insights on legitimate, questionable and anomalous usages detected during this first iteration of the measurement campaign.

5.1 Enrolment timeline

This section concentrates on the 181 domains that are present in both the *Attested* and *Allowed* sets. By processing the JSON attestation file described in Section 3.4, we can visualise the onboarding process over time by extracting the issue date from each attestation file. Figure 5.1 shows a timeline of the onboarding, which counts the number of enrolments on a daily basis.

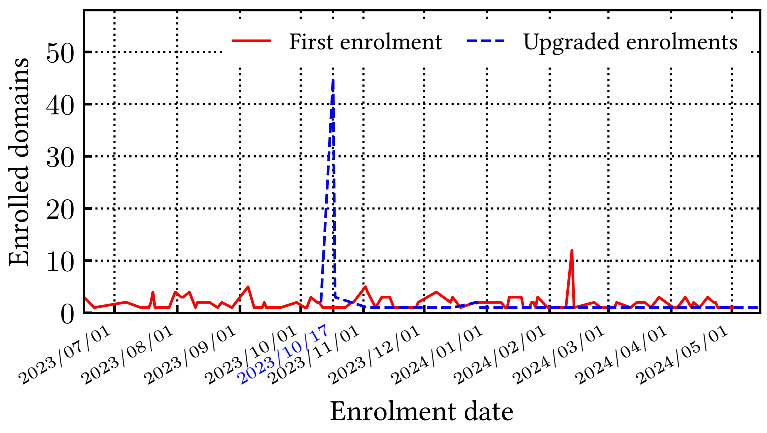


Figure 5.1: Timeline showing the enrolment date of third parties that are present in both the *Attested* and *Allowed* sets. In red, the number of third-party services that have received the attestation file for the first time. In blue, the number of services that have renewed the enrolment from the second time onwards.

We can deduce from the figure that the enrolments kicked off in June 2023, with the first attestations being generated on the 16th. Until May 2024, the enrolment

[1]This domain is a different case from `dstillery.com`: by analysing its attestation file, we notice that the `enrollment_site` is `tailtarget.com`, which is in the *Allowed* set. This likely means that the two domains are registered under the same company. On the other hand, Dstillery's domain and the `enrollment_site` in its attestation file match.

process proceeds at a very slow pace, with only approximately a dozen new domains obtaining an attestation file per month. Curiously, the timeline shows a peak of updates to the attestation files on October 17th, 2023. Upon further analysis, we notice that this peak coincides with the update of the attestation version, which included the addition of the `enrollment_site` field and other minor changes. This update was performed by 45 third parties, out of the 61 total that were already enrolled—and thus, already had an attestation file—before that date.

5.2 Legitimate usage

In this section, we concentrate on the *legitimate* uses of the Topics API. Therefore, we only consider the invocations that were performed by the 47 CPs that are both present in the *Allowed* and *Attested* sets, only on those websites where the crawler was able to accept the privacy policy (D_{AA} dataset). In particular, we focus on what are the most popular advertisement platforms that adopt the Topics API.

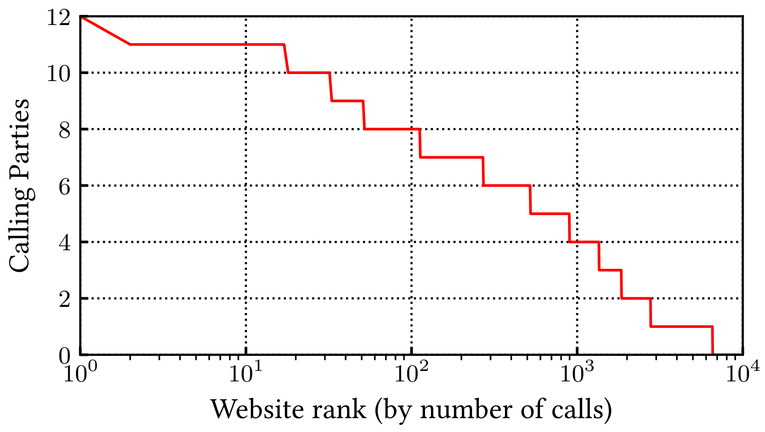


Figure 5.2: Distribution of the Topics API calls among the websites visited in *After-Accept*. The graph counts the amount of CPs found within each website, sorted by this number in descending order.

First, we analyse how the Topics API calls are distributed among the visited websites: Figure 5.2 counts the number of CPs that invoked the Topics API for each visited website. It appears that 12 is the most CPs embedded by a single website, with the other top-10 websites at 11. The top-100 websites by number of embedded CPs host 8 or more, while the top 1,000 hosts 4 or more. From this graph, we can observe around 6,500 websites with at least one call to the Topics API: this is around 45% of the total 14,719 websites visited in *After-Accept*, meaning that almost one every two websites hosts a legitimate call to a CP. This hints that the authorised third parties are already showing interest towards the new technology.

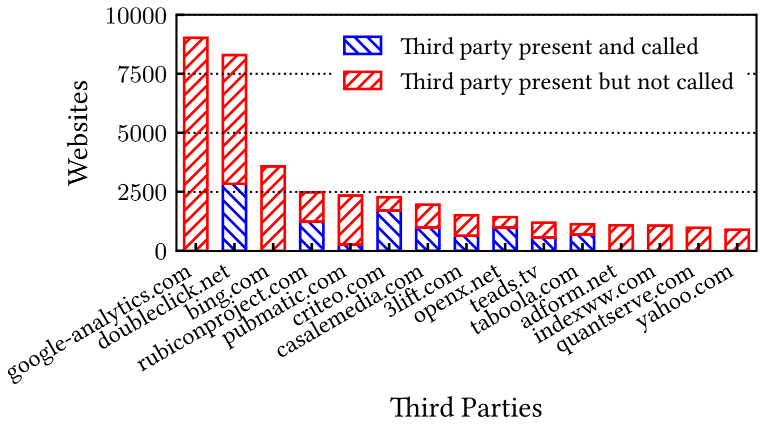


Figure 5.3: Number of websites where *Allowed* and *Attested* third-party services were present, and the subset where a call to the Topics API was recorded, sorted in descending order by the number of websites where the third-party service was found. Dataset D_{AA} is considered for this graph.

Next, we move onto the third-party services, discovering which ones among the most popular tend to invoke the Topics API. Figure 5.3 details the number of websites where the most popular *Attested* and *Allowed* third parties are present, shown in red. In blue, we highlight the subset of websites where the same third party also invokes the Topics API, and can therefore be considered a CP. It appears that the top-10 third parties, all of which are well-known, perform some calls, except for two of them—namely, `google-analytics.com` and `bing.com`. This is logical, as both Google Analytics and Bing are not ad-related services. The CPs that appear to perform the most calls are `doubleclick.net`, `criteo.com`, `rubiconproject.com` and `casalemedia.com`. However, none of these CPs appear to invoke the Topics API on every website. On the contrary, calls seem to appear with different levels of frequency for each CP: `criteo.com` performs a call on about three websites every four it is embedded in, `rubiconproject.com` and `casalemedia.com` on about half of them, while `doubleclick.com` only on about one third of them. In general, these results show that the most major ad-related third parties have started employing the Topics API in their own production environment, although not consistently. This is a clear hint that there is some ongoing testing process from most of the observed platforms.

An effective way to investigate this aspect is by further examining the frequency with which the various CPs call the Topics API on the websites they appear in, that

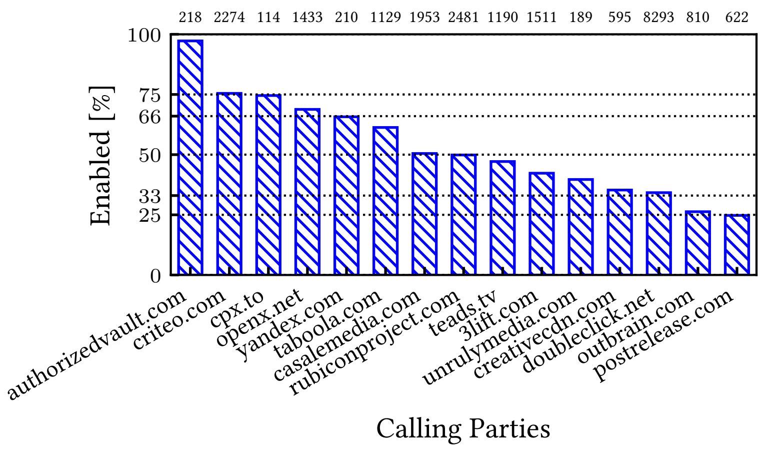


Figure 5.4: Top-15 *Allowed* and *Attested* CPs with the largest enabled percentage. The enabled percentage is the percentage of times a CP performs a call to the Topics API over the total number of websites it appears in (shown in the top row), sorted in descending order by this value. For a clearer result, only the most meaningful CPs were considered, i.e., those that were found in at least 100 websites. The dataset considered for this graph is D_{AA}.

is the ratio between the amount of calls recorded from a given CP and the total number of websites it appears in. Figure 5.4 shows the CPs with the highest enabled percentages, which highlights some fractions on the y-axis to simplify reading the results and details the total number of times the CP is observed on the top. We can notice some interesting behaviours: for instance, `authorizedvault.com`, present on 218 websites, is the only CP that calls the Topics API amost every time. In contrast, `criteo.com` and `cpx.to` perform a call around 75% of the time, `yandex.com` around 66% of the time, `casalemedia.com` and `rubiconprpject.com` almost exactly 50% of the time and so on. This pattern of percentages further demonstrates that there probably is some sort of A/B testing going on with these platforms, with percentages that look predetermined. This process can be highly beneficial for companies, as it would allow them to compare the effectiveness of the Topics API paradigm with that of the current one based on third-party cookies for their business metric.

To further verify the occurrence of A/B testing, we run repeated tests on select websites to observe the policy CPs use to enable or disable the Topics API. Specifically, we perform 100 consecutive visits to 5 websites for each of the top-10 CPs shown in Figure 5.4, with the websites manually selected based on the presence of the respective CP. We notice consistent alternative periods: for some time, CP, and website, the usage of the API appears to be called for all visits, followed by some time when it appears disabled. This is consistent with common patterns of A/B testing, where the CP considers the same population and websites but at different times.

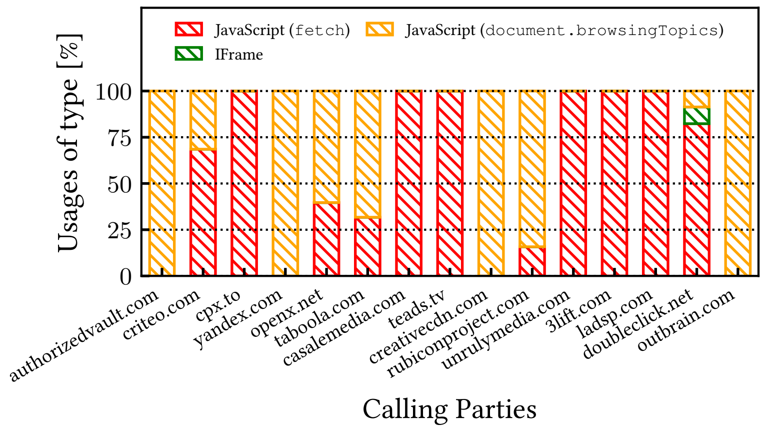


Figure 5.5: Distribution of Topics API usages in D_{AA}, separately by CP. The CPs shown are the same displayed in Figure 5.4.

Next, we concentrate on what usage types (introduced in Section 3.5) are the most prominent among the various CPs. Considering only the *Attested* and *Allowed* CPs in D_{AA}, we find that the JavaScript and Fetch usage types are used about equally, accounting for 47.6% and 48.8% of total calls, respectively. In contrast, the Iframe usage type is the least common, found in only 3.6% of calls. Interesting results emerge if we compute this percentage separately by CP. Figure 5.5 shows these percentages, calculated for each CP that appears in Figure 5.4. We find some CPs that exclusively adopt one of the types between JavaScript and Fetch, while others such as `criteo.com`, `openx.net` and `taboola.com` seem to alternate between the two types. The figure also highlights how `doubleclick.net` appears to be the only one among the 15 listed that adopts all three usage types. This is somewhat expected, given that DoubleClick has been acquired by Google, the creators of the Topics API. These results suggest that different CPs are experimenting with different usage types, experimenting on which would be more effective for them. Overall, these patterns show even further evidence that most Topics API adopters are conducting some sort of A/B testing, where the usage type represents another parameter being taken into account.

5.3 Anomalous usage

In this section, we concentrate on the 2,615 CPs in D_{AA} that access the Topics API despite not being in the *Allowed* set. The presence of these domains arises two main questions:

- How did over 2,000 unauthorised CPs successfully perform a Topics API call if, as explained in Section 3.4, the browser is supposed to block all calls from them? We found an implementation error in Chromium's code that allows a CP to bypass the browser's *Allowed* domain check, further detailed in Section 5.3.1.
- Who are the CPs that perform these anomalous calls? Is there an underlying reason behind these calls? We attempt to answer these questions in Section 5.3.2.

5.3.1 Bypassing Chromium's check for the *Allowed* domains

While inspecting Chromium's code, we found a function called `IsSiteAttested`[2] that behaves as shown in Algorithm 1. Basically, if the file is absent or corrupted, and a specific browser feature is enabled, then the site will be automatically considered *Allowed*. This feature is enabled by default in a fresh browser installation. Therefore, if a possibly malicious entity finds a way to delete or corrupt the allow-list, it would allow any caller to access the API, regardless of whether *Allowed* or not, permitting them to collect the user's topics and possibly abuse this information.[3]

Moreover, we have noticed that the browser does not download the allow-list right after it's opened, but it takes several minutes. In some cases, the list was not downloaded until after having closed and re-opened the browser. In any case, the crawling of each website consisted was peformed independently within separate disposable Docker containers [48], which were re-created for each visited website. Given that, on average, the crawling of a single website takes only about two minutes, the browser likely didn't have sufficient time to download the `privacy-sandbox-attestations.dat` list before the crawl finished, and the

[2]The source code of this function can be viewed at `https://source.chromium.org/chromium/chromium/src/+/refs/tags/122.0.6261.128:components/privacy_sandbox/privacy_sandbox_attestations/privacy_sandbox_attestations.cc;l=275`

[3]The actual feasibility of the attack goes beyond the scope of this thesis. At the moment of writing, Google and Chromium developers have been notified about the error. They acknowledged the problem and declared to fix it in a future release. At the moment of writing, the bug is still present in Chrome and Chromium's latest version.

Algorithm 1 Simplified logic of the `IsSiteAttested` function.

```
function IsSiteAttested(site, ...)
    status, allowList ← ReadAllowList(...)
    if status is FileNotPresent or FileCorrupted then
        if feature DefaultAllowPrivacySandboxAttestations is enabled then
            return Allowed
        else
            return status
        end if
    else
        if site in allowList then
            return Allowed
        else
            return Denied
        end if
    end if
end function
```

relative container was disposed of. As a result, the crawler was able to record the calls from *every* CP instead of only from the ones in the *Allowed* set.

5.3.2 Identifying the sources and causes of anomalous calls

While investigating the nature of the anomalous calls, we observe that 72% out of the 3,450 total anomalies was generated from the same website being visited, meaning that the website and the CP's 2LD coincide (e.g., `www.website.com` and `ad.website.net`). A manual check on the remaining 28% reveals similar situations, where i) the same company owns the two domains (e.g. `windows.com` and `microsoft.com`) or ii) the visited website redirects to a second website which then calls the API, where both websites are owned by the same company.

Additionally, we observe that *all* anomalous calls were performed through JavaScript's `browsingTopics` function, hinting at one or more popular JavaScript libraries erroneously implementing the API call. If these libraries were downloaded by the browser from an external source but placed directly within the website's page (e.g., a `<script>` tag with a `src` attribute), the relative calls would indeed result as originating not from the script's source but from its browsing context [41] which, in this case, would be the website itself. Figure 5.6 clarifies the difference between the browsing context and the script's source.

To find a possible culprit, we observe on 95% of the websites containing an anomalous call the presence of GTM (Google Tag Manager). By manually inspecting

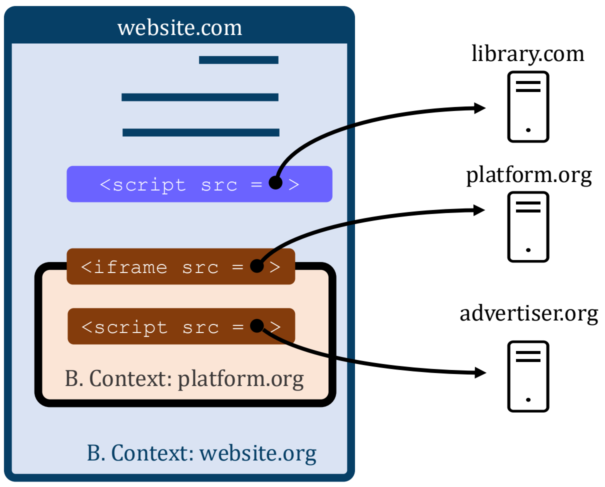


Figure 5.6: Example showing the difference between a browsing topic and a script's source. In this example, two scripts are downloaded from the browser: one is part of the HTML code of `website.com`, while the other is embedded inside an iframe that downloads content from `platform.org`. If both scripts contain a call to `browsingTopics`, the corresponding Topics API call from the one downloaded from `advertiser.org` will have `platform.org` as its browsing context, whereas the call that originates from `library.com`'s script will appear as incoming from `website.com`.

its source code we notice that it, indeed, contains a call to the `browsingTopics` function. This is rather strange, as GTM is neither *Attested* nor *Allowed*.

In websites where GTM is present, the `<script>` tag is included directly within the HTML of their page. Whenever a user connects to one of these websites, the browser downloads a script from a link similar to `https://googletagmanager.com/gtm.js?id=<ID>`. When the `browsingTopics` function is reached, the call occurs, but since the script is executed within the *root* browsing context, the call's context origin will be set to the website instead of GTM. This is the exact scenario shown in Figure 5.6, where `library.com` is replaced with GTM's domain.

GTM is the most prominent example of this issue, but likely not the only one: the remaining 5% of anomalous calls detected were probably triggered from other, less popular libraries that made the same mistake. This problem is relevant for both the Topics API itself, as it could pose complications for its deployment,[4] but also websites that implement third parties such as GTM, as they may potentially

[4]We have contacted Google about this issue as well, but at the moment of writing we did not receive any response.

cause unexpected and unwanted privacy issues.

5.4 Questionable usage

This final section will concentrate on the Topics API calls that were performed during the *Before-Accept* visit, thus examining the D_{BA} dataset. Since we performed the measurements from Europe, where the citizens are protected by the GDPR, we would expect no usage of the Topics API in this stage of the crawling, not having yet accepted any privacy policy. Yet, we find over 1,300 CPs that have performed at least one, out of which only 28 are *Allowed*. These usages are considered questionable and can be seen as a violation of European regulations, since the disclosure of topics can be seen as equivalent to third-party cookies.[5]

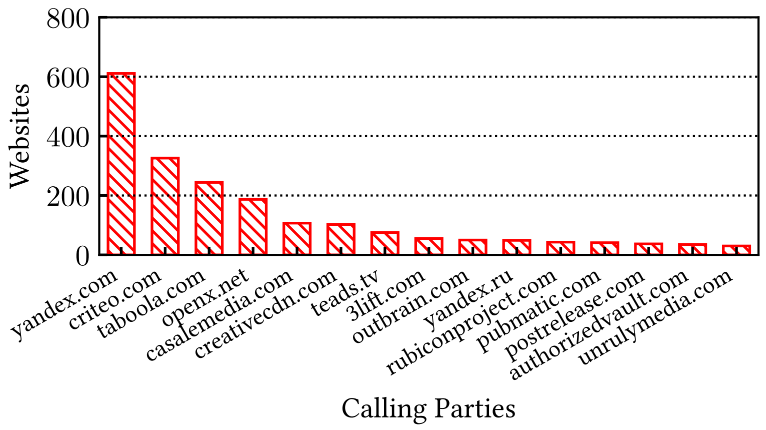


Figure 5.7: Number of websites where a Topics API call was recorded in *Before-Accept*, for each *Allowed* CP. The graph shows this statistic for the top-15 CPs with the highest count.

Concentrating on the 28 *Allowed* CPs that invoke the Topics API in *Before-Accept*, Figure 5.7 illustrates the top-15 CPs that perform a questionable call. We observe that `yandex.com` performs the largest amount of these calls (611 in *Before-Accept*), despite not being among the top callers overall (1,414 in *After-Accept*). Although we do find some of the major callers in this graph (e.g., `criteo.com`), there seems to be little correlation with the service's popularity. For instance, `doubleclick.net`, the top caller in *After-Accept* as shown in Figure 5.3, does not

[5]This thesis will not discuss whether this could be considered an actual violation of the GDPR, but the fact that some services respect this interpretation reinforces these claims.

perform any call in *Before-Accept*. This further validates the expectation that no calls should be issued during *Before-Accept* visits.

There are two main cases that can justify this behaviour:

- The website is hosted outside the European Union and does not include any privacy banner, thereby allowing any privacy-invasive technology in every visit. This aspect would still account as a GDPR violation, which protects European citizens even when accessing services outside the EU. We discuss the correlation between questionable calls and the geographical region of a website in Section 5.4.1.

- The website does not correctly implement their privacy banner. Most websites tend to adopt the use of a CMP (Consent Management Platform) to handle the technical aspects of web privacy. However, these platforms often need to be configured by the website administrator, and an incomplete configuration could lead to potential violations of privacy regulations. The correlation between questionable calls and privacy banner misconfiguration is discussed in Section 5.4.2.

5.4.1 Correlation with the websites' geographical region

In this section, we investigate a possible correlation between the presence of a questionable call and the geographic location of a website. We do this by checking the TLD (Top-Level Domain) of the websites where we observe a questionable call, as the TLD (Top-Level Domain) of a website can give a rough estimation of its country. Figure 5.8 highlights the differences of questionable call frequency between the most relevant TLDs, effectively breaking down questionable calls by geographic region: international (`.com`), Japan (`.jp`), Russia (`.ru`), the EU (30 TLDs of countries where the GDPR is in force), and the remaining less popular TLDs. We can first observe that the presence of CPs strongly varies in different regions. For example, Yandex, a Russian company, is unsurprisingly very prevalent in Russia, while being almost absent in the EU and not appearing at all in Japan. On the other hand, Criteo, based in France, has more of a global presence. The same stands for Taboola and OpenX, both based in the United States, although in smaller numbers compared to Criteo. While this difference in deployment can be caused by different strategies from websites and the advertisement companies, the graph shows no significant correlation between the frequency of questionable calls and the geographical region. Interestingly, we even observe several questionable calls within websites in the EU, where we would definitely expect no questionable calls originating from.

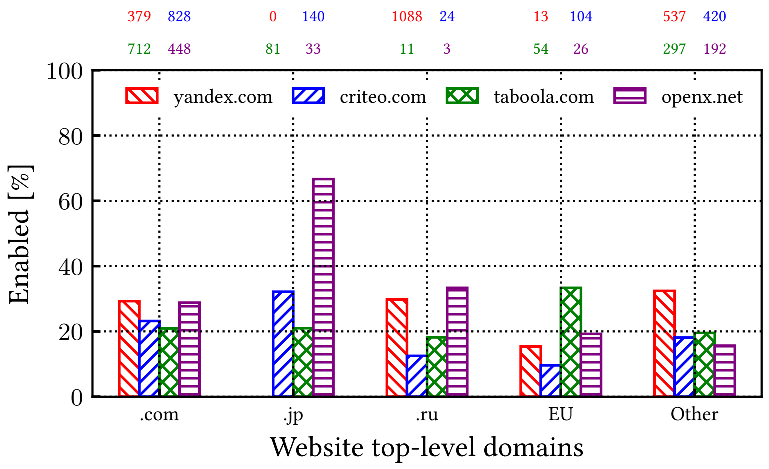


Figure 5.8: Percentage of websites where a Topics API call was recorded in *Before-Accept*, separately for each of the most relevant classes of TLDs and for the top-4 questionable CPs shown in Figure 5.7. The numbers on the top show the total amount of websites of the given TLD where the CP was found.

5.4.2 Correlation with improper privacy banner configuration

Since the analysis of geographical regions did not give significant results, we next investigate whether this questionable behaviour is influenced by the incorrect configuration of privacy banners by the website administrators. To this end, we try to paint a picture of the presence of CMPs (Consent Management Platforms) within websites where a questionable call occurs. CMPs are commercial products which simplify the implementation of privacy banners and offer libraries that control the third parties embedded within the websites (such as advertisers or trackers), blocking them until the user has consented to the Privacy Policy. These plugins require minimal configuration from the website administrator, through either code or a control panel. If this is incomplete, the CMP may block only some—or even none—of the third-party trackers before the user has provided consent (i.e., in the *Before-Accept* visit), thus violating the GDPR [23]. We assume that the Topics API should follow the same regulations as other privacy-intrusive features such as third-party cookies. Therefore, the presence of a Topics API call during the *Before-Accept* visit could be due to incorrect CMP configuration, or even implementation errors on the CMP's part.

The information about the presence of CMPs within a website can be retrieved from *Priv-Accept*'s output, scanning for the domain names of the most popular CMPs within the contacted third parties. These domains are then mapped to the

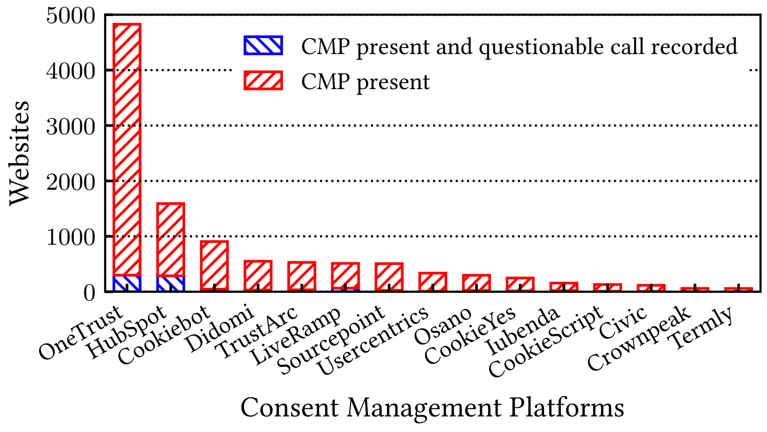


Figure 5.9: Number of total websites where the top-15 most popular CMPs are present during the *Before-Accept* visit, and the subset of websites where a questionable call was recorded.

corresponding CMP, thanks to the list offered by Wappalyzer [49]. Figure 5.9 shows the most prevalent CMPs encountered during the *Before-Accept* visits, highlighting the number of websites where a questionable Topics API call was recorded. We observe that OneTrust is the most prevalent CMP by far, appearing in almost 5,000 websites. HubSpot and CookieBot are also adopted quite often, from around 1,500 and 1,000 websites, respectively. However, we observe that only OneTrust and HubSpot have a significant amount of websites that witness a questionable call, at around 300 each. This sparks some suspicion: given the difference in number of appearances between OneTrust and HubSpot, we would ideally expect a similar difference with the number of recorded questionable calls. However, this does not seem to be the case, as they appear similar. In other words, we would expect the probability of encountering a certain CMP over any website, which we will indicate as $P(\text{CMP} = x)$, to be independent of the probability of witnessing a questionable Topics API call, indicated as $P(\text{questionable call})$. On the contrary, we find evidence that there may be some correlation between the two events.

We examine the likeliness of this correlation by comparing separately for each CMP the value of $P\,(\text{CMP} = x)$ with the probability of observing the same CMP within a website where a questionable call was recorded, indicated as $P\,(\text{CMP} = x \mid \text{questionable call})$. Figure 5.10 shows this comparison. In an ideal scenario, we would expect the two values to be equal. While many of the CMPs show very similar probabilities, we do find some odd cases. Primarily, HubSpot shows a probability of being the CMP in use given a questionable call which is about 3 times the probability of observing it, hinting towards a possible mishandling of

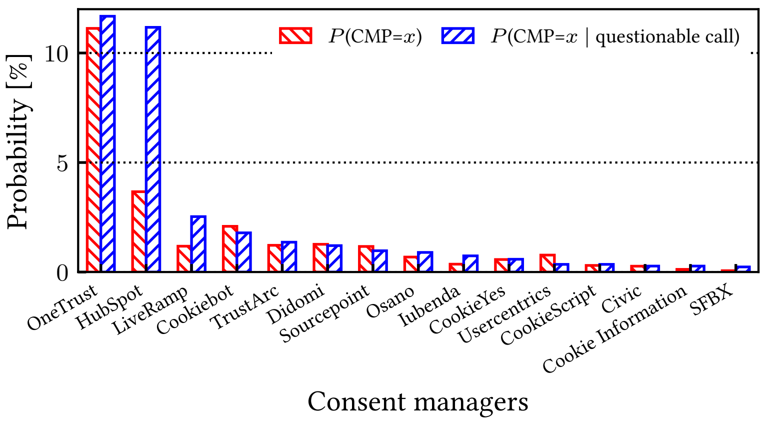


Figure 5.10: Comparison of the probability to encounter a given CMP (in red) and the probability to encounter that CMP in a website where a questionable call was recorded (in blue), for the CMPs with the highest value of $P\,(\text{CMP} = x \mid \text{questionable call})$.

the Topics API. The same holds true for LiveRamp.

Overall, it is likely that CMPs have yet to properly integrate the support for the Topics API, leading to possible violations of privacy regulations.

5.5 Longitudinal measurements

In this section, we analyse the results of the same measurement campaign performed over an extended period of time, on a weekly basis. This analysis aims to provide a clearer idea of the evolution of the Topics API's deployment over time. The measurements were taken starting at midnight on every Monday, over the top 50,000 websites according to the latest version of the Tranco list at the time of each crawl. We discuss the results collected from August 12th to October 14th, 2024, totaling 9 weeks of measurements. We could not perform the crawling on August 26th due to technical malfunctions.

First, we concentrate on the *Allowed* third-party services which can be found in the `privacy-sandbox-attestations.dat` allow-list, by examining how the number of these services changes over time and highlighting the presence of new potential adopters of the Topics API. Figure 5.11 tracks this number over the nine-week measurement period, while also showing how many of these parties are also *Attested*, effectively providing a time-based view of the values present in the first group of rows in Table 5.1. The graph shows that the number of *Allowed* services remain relatively stable during the period in question. However, it also

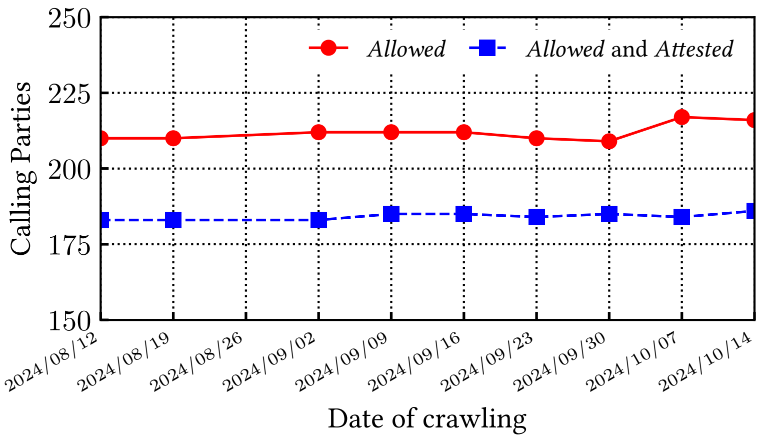


Figure 5.11: Evolution of the *Allowed* and *Attested* third parties collected over the 9 weeks of measurements from the browser's local allow-list.

shows an increase of total domains in the allow-list since March: we observe over 200 *Allowed* third parties in August, compared to 193 in March. Additionally, the graph shows that some third parties remain *Allowed* but not *Attested*, represented by the interval between the two lines. We notice that this amount has increased since March: while the number of *Allowed* domains has increased to 216 according to the latest measurements, the number of *Allowed* and *Attested* domains remains closely the same, recorded at 186. Therefore, the 12 *Allowed* and not *Attested* parties found in March have more than doubled during this period, reaching a total of 30. These results suggest that new third-party services are being added to the local browser allow-list, despite not having correctly exposed the JSON attestation file witihin their domain.

We now focus on what websites witness Topics API calls, to answer whether *Allowed* and *Attested* third parties are increasingly adopting the new technology. We achieve this by plotting the changes in percentage of websites that witness at least one legitimate CP—i.e., an *Allowed* and *Attested* third party that calls the Topics API during the *After-Accept* visit—shown in Figure 5.12. We observe this percentage oscillating between 39% and 42%, therefore remaining somewhat stable. However, these values appear slightly lower compared to the 45% encountered in March. This difference is probably due to the variance in websites in the Tranco list, especially towards the bottom: the websites visited in August onwards may have been very different from the ones visited in March.

Next, we examine the variation in the number of legitimate, questionable, and anomalous usages over time, as described in the previous sections of this chapter. These changes are highlighted in Figure 5.13. While the evolution of these numbers

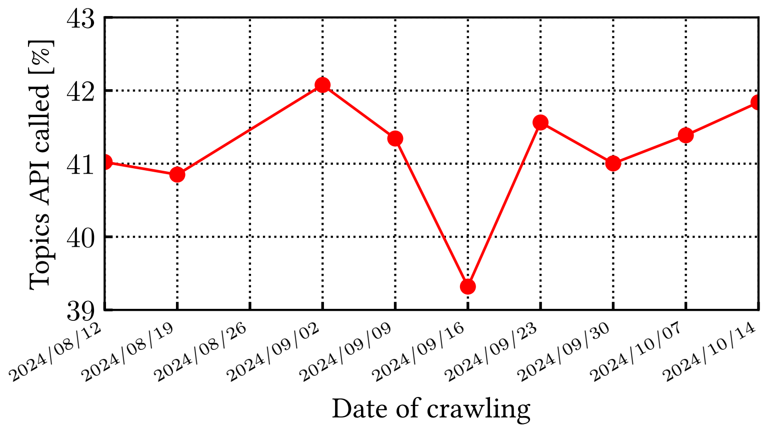


Figure 5.12: Evolution of the percentage of websites where at least one legitimate Topics API call occurs. A legitimate call refers to those done by an *Allowed* and *Attested* third-party during the *After-Accept* visit.

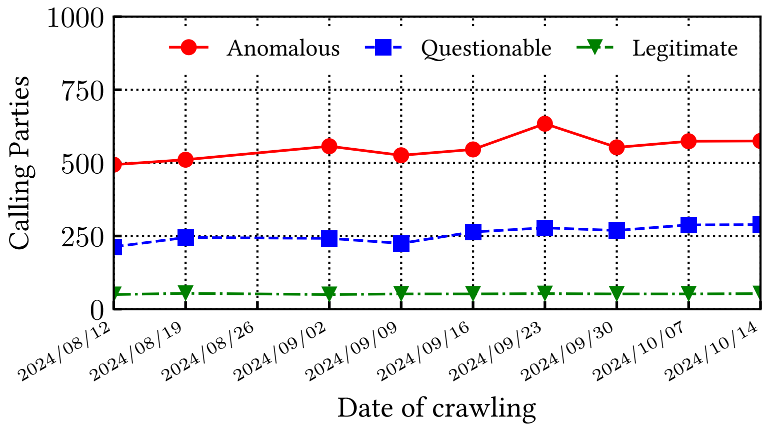


Figure 5.13: Evolution of the number of legitimate, questionable and anomalous CPs found over the nine-week measurement period. Legitimate CPs are those that are both *Allowed* and *Attested* and invoked the Topics API during the *After-Accept* visit. Questionable CPs refer to those that called the API during the *Before-Accept* visit. Anomalous CPs are those that invoked the Topics API in the *After-Accept* visit but are not *Allowed* or *Attested*.

appears relatively stagnant for all three categories, it seems that the amount of questionable and anomalous CPs has dropped significantly since March. In fact, we observe between 500 and 650 anomalous CPs within the graph, while Table

5.1 shows over 2,600 of them in March. This suggests that one or more popular libraries may have updated their implementation to address the issues described in Section 5.3.2 between March and August. Questionable CPs seem to follow a similar trend, as we witness a drop from over 1,300 to just around 250. This may also suggest that one or more CMPs have addressed their implementation issues regarding the Topics API during this time. As for legitimate CPs, we notice very few changes in their numbers, even compared to March's measurements, which remain constant at around 50 callers.

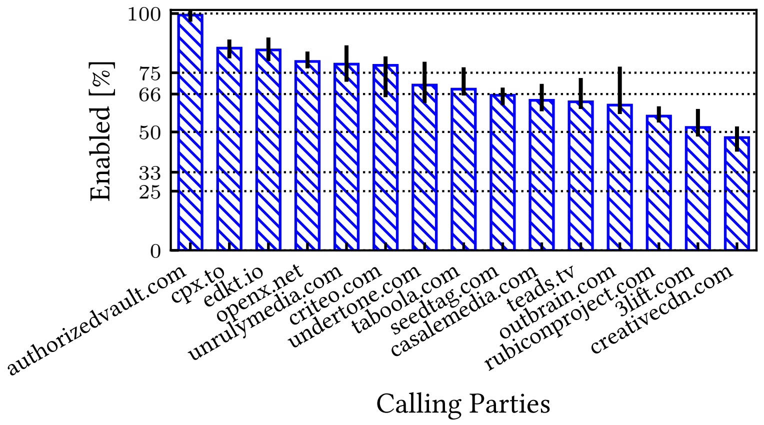


Figure 5.14: Top-15 CPs with the largest average enabled percentage. The enabled percentage is the percentage of times a CP performs a call to the Topics API over the total number of websites it appears in. The black bars highlight the range, from minimum to maximum, of that percentage encountered over the nine-week measurement period. Like in March's measurements, this graph only shows the *Allowed* and *Attested* CPs that are found in at least 100 websites.

Finally, we review whether the enabled percentage during the *After-Accept* visit, previously displayed in Figure 5.4, is increasing over time, which would indicate progress in the A/B testing process. Figure 5.14 highlights the CPs with the highest average enabled percentage, while also highlighting the interval of percentages recorded over time. Compared to March, these percentages appear to have increased on average, with low fluctuation over time. We also observe some new callers, such as `edkt.io`, `undertone.com` and `seedtag.com`. `authorizedvault.com` remains the only significant CP that calls the Topics API on every website it appears in, whereas most of the others still appear to stop at what look like predefined values, hinting that the A/B testing is still ongoing.

Chapter 6

Conclusions

This thesis painted a comprehensive picture of the Topics API's current status of deployment, thanks to a measurement campaign conducted with an enhanced version of an existing Web crawler. This tool allowed us to record the Topics API usages from third parties during website visits, distinguishing between calls made before and after a user has provided consent to a website's privacy policy. We explored a snaphot of the API's deployment based on a single day's measurements, as well as multiple measurements performed over time to track its evolution. The collected datasets allowed us to identify which advertising platforms, if any, have begun experimenting with the Topics API as a tool to facilitate IBA. The collected information also helped us determine the presence of illegitimate calls, such as those performed before the user has accepted a website's Privacy Policy, thereby violating European regulations.

The results show that most major advertising platforms are already deploying and experimenting with the Topics API in the real world, given the now precarious position of third-party cookies. The total number of platforms adopting the new technology, according to the collected longitudinal measurements, appears to be growing slowly but steadily. However, they also show clear signs that this experimentation is being performed in form of A/B tests on controlled subsets of websites and users. The measurements taken over time suggest that this process is still ongoing.

Interestingly, we found a significant number of *questionable* calls from websites and third-parties before the user has accepted the Privacy Policy, that hint at a wrong implementation of privacy banners and CMPs. Since this could lead to violations of regulations such as the GDPR, this thesis serves as a reminder for website administrators to verify that their privacy banners are properly configured, as well as for CMPs to carefully evaluate their implementation of the Topics API within their libraries.

We also find evidence of erroneous implementations of advertising platforms and libraries, resulting in *anomalous* calls. These cases were discovered thanks to an implementation bug found in Chromium's Topics API implementation. We estimate the underlying source of anomalous calls to be the erroneous implementation of JavaScript libraries such as GTM, a problem which developers need to be made aware of.

On a positive note, both numbers of questionable and anomalous calls seem to be decreasing over time, hinting that the mentioned configuration and implementation errors are being addressed by at least some of the responsible parties.

Being proposed by one the largest tech giants, the Topics API has the potential to become the *de facto* standard for IBA in the future. However, this research has shown that the technology still exhibits clear signs of being in its early stages of development. Such signs include bugs and unexpected behaviour, of which all stakeholders in the system—advertisers, consumers, privacy advocates and even Google itself—need to be aware of. Furthermore, the general approval of this technology is still uncertain, as the interests of advertisers, mainly based around building extensive and fine-grained user profiles, are in contrast with the restrictions imposed by the Topics API. Overall, the future of this technology remains hard to predict.

6.1 Limitations and future work

The tools built during this thesis work represent an initial effort of understanding the deployments of this new technology. However, at the moment of writing, they have several limitations that present opportunities for future research:

- First, the collected measurements only record the calls performed to the Topics API, without providing information about their purpose. Future research could explore how advertisers utilise the retrieved topics to, for instance, provide different ads based on the disclosed interests.
- Second, the measurements only cover a small time frame of 2024. Given the early stages of this technology, conducting the same campaign for longer periods of time would offer a more comprehensive view of its evolution over time.
- Finally, the measurements were collected from a single location in Europe, which may be limiting. An interesting topic of future research could involve gathering the same measurements from different geographical regions, to determine if any significant changes appear in the technology's behaviour.

Appendix A

Modified Chromium Source code

```
bool BrowsingTopicsSiteDataStorage::CreateSchema() {
    static constexpr char kBrowsingTopicsApiUsagesCompleteTableSql
   [] =
        "CREATE TABLE IF NOT EXISTS
   browsing_topics_api_usages_complete("
            "hashed_context_domain INTEGER NOT NULL,"
            "context_origin_url TEXT NOT NULL,"
            "hashed_main_frame_host INTEGER NOT NULL,"
            "caller_source TEXT NOT NULL,"
            "usage_time INTEGER NOT NULL,"
            "PRIMARY KEY (context_origin_url,
   hashed_main_frame_host,usage_time))";
    if (!db_->Execute(kBrowsingTopicsApiUsagesCompleteTableSql))
        return false;

    static constexpr char kBrowsingTopicsApiUsagesViewSql[] =
        "CREATE VIEW IF NOT EXISTS browsing_topics_api_usages AS "
            "(SELECT hashed_context_domain,"
                "MAX(usage_time) AS last_usage_time,"
                "hashed_main_frame_host "
            "FROM browsing_topics_api_usages_complete "
            "GROUP BY hashed_context_domain,
   hashed_main_frame_host)";
    if (!db_->Execute(kBrowsingTopicsApiUsagesViewSql))
        return false;

    static constexpr char kLastUsageTimeIndexSql[] =
        "CREATE INDEX IF NOT EXISTS usage_time_idx "
            "ON browsing_topics_api_usages_complete(usage_time)";
    if (!db_->Execute(kLastUsageTimeIndexSql))
```

```
        return false;

    static constexpr char kHashedToUnhashedDomainSql[] =
        "CREATE TABLE IF NOT EXISTS "
            "browsing_topics_api_hashed_to_unhashed_domain("
            "hashed_context_domain INTEGER PRIMARY KEY,"
            "context_domain TEXT NOT NULL)";
    if (!db_->Execute(kHashedToUnhashedDomainSql)) {
        return false;
    }

    return true;
}
```

Listing A.1: Modified code of BrowsingTopicsSiteDataStorage's CreateSchema function.

```
void BrowsingTopicsSiteDataStorage::OnBrowsingTopicsApiUsed(
    const browsing_topics::HashedHost& hashed_main_frame_host,
    const browsing_topics::HashedDomain& hashed_context_domain,
    const std::string& context_origin_url,
    const std::string& caller_source,
    const std::string& context_domain,
    base::Time time) {
    DCHECK_CALLED_ON_VALID_SEQUENCE(sequence_checker_);

    if (!LazyInit())
        return;

    sql::Transaction transaction(db_.get());
    if (!transaction.Begin())
        return;

    static constexpr char kInsertApiUsageSql[] =
        // clang-format off
        "INSERT OR REPLACE INTO
  browsing_topics_api_usages_complete "
            "(hashed_context_domain,hashed_main_frame_host,
  usage_time,context_origin_url,caller_source) "
            "VALUES (?,?,?,?,?)";
    // clang-format on

    sql::Statement insert_api_usage_statement(
                        db_->GetCachedStatement(SQL_FROM_HERE,
  kInsertApiUsageSql));
    insert_api_usage_statement.BindInt64(0, hashed_context_domain.
  value());
    insert_api_usage_statement.BindInt64(1, hashed_main_frame_host
  .value());
```

```
    insert_api_usage_statement.BindTime(2, time);
    insert_api_usage_statement.BindTime(3, context_origin_url);
    insert_api_usage_statement.BindTime(4, caller_source);

    if (!insert_api_usage_statement.Run()) {
        return;
    }

    static constexpr char kInsertUnhashedDomainSql[] =
        // clang-format off
        "INSERT OR REPLACE INTO
   browsing_topics_api_hashed_to_unhashed_domain "
            "(hashed_context_domain,context_domain) "
            "VALUES (?,?)";
    // clang-format on
    sql::Statement insert_unhashed_domain_statement(
        db_->GetCachedStatement(SQL_FROM_HERE,
   kInsertUnhashedDomainSql));
    insert_unhashed_domain_statement.BindInt64(0,
   hashed_context_domain.value());
    insert_unhashed_domain_statement.BindString(1, context_domain)
   ;

    if (!insert_unhashed_domain_statement.Run()) {
        return;
    }

    transaction.Commit();
}
```

Listing A.2: Modified code of BrowsingTopicsSiteDataStorage's OnBrowsingTopicsApiUsed function.

```
bool BrowsingTopicsServiceImpl::HandleTopicsWebApi(
    const url::Origin& context_origin,
    content::RenderFrameHost* main_frame,
    ApiCallerSource caller_source,
    bool get_topics,
    bool observe,
    std::vector<blink::mojom::EpochTopicPtr>& topics) {

    // [...]

    std::string context_origin_url = context_origin.GetURL().spec
   ();

    std::string caller_source_str;
    switch(caller_source) {
        case ApiCallerSource::kJavaScript:
```

```
        caller_source_str = "javascript";
        break;
        case ApiCallerSource::kFetch:
        caller_source_str = "fetch";
        break;
        case ApiCallerSource::kIframeAttribute:
        caller_source_str = "iframe";
        break;
        default:
        caller_source_str = "invalid";
        break;
    }

    std::string context_domain =
        net::registry_controlled_domains::GetDomainAndRegistry(
            context_origin.GetURL(),
            net::registry_controlled_domains::
INCLUDE_PRIVATE_REGISTRIES);

    HashedDomain hashed_context_domain =
HashContextDomainForStorage(
        browsing_topics_state_.hmac_key(), context_domain);

    if (observe) {
        // Track the API usage context after the permissions check
.
        BrowsingTopicsPageLoadDataTracker::GetOrCreateForPage(
main_frame->GetPage())
            ->OnBrowsingTopicsApiUsed(caller_source_str,
context_origin_url,
                                    hashed_context_domain,
context_domain,
                                    history_service_);
    }

    // [...]
}
```

Listing A.3: Modified code of `BrowsingTopicsServiceImpl`'s `HandleTopicsWebApi` function.

Bibliography

[1] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. «Detecting and Defending Against Third-Party Tracking on the Web». In: *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. San Jose, CA: USENIX Association, Apr. 2012, pp. 155–168. ISBN: 978-931971-92-8. URL: `https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/roesner` (cit. on pp. 2, 6).

[2] 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*. CCS '16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 1388–1401. ISBN: 9781450341394. DOI: `10.1145/2976749.2978313`. URL: `https://doi.org/10.1145/2976749.2978313` (cit. on pp. 2, 6).

[3] Balachander Krishnamurthy and Craig Wills. «Privacy diffusion on the web: a longitudinal perspective». In: *Proceedings of the 18th International Conference on World Wide Web*. WWW '09. Madrid, Spain: Association for Computing Machinery, 2009, pp. 541–550. ISBN: 9781605584874. DOI: `10.1145/1526709.1526782`. URL: `https://doi.org/10.1145/1526709.1526782` (cit. on pp. 2, 6).

[4] *AdBlock Plus*. `https://adblockplus.org`, accessed on 2024-10-17. 2024 (cit. on p. 2).

[5] *Ghostery*. `https://www.ghostery.com`, accessed on 2024-10-17. 2024 (cit. on p. 2).

[6] *Disconnect*. `https://disconnect.me`, accessed on 2024-10-17. 2024 (cit. on p. 2).

[7] *Brave*. `https://brave.com/`, accessed on 2024-10-17. 2024 (cit. on p. 2).

[8] *DuckDuckGo*. `https://duckduckgo.com/`, accessed on 2024-10-17. 2024 (cit. on p. 2).

[9] European Parliament and Council of European Union. *Directive 95/46/EC. General Data Protection Regulation.* `http://data.consilium.europa.eu/doc/document/ST-5419-2016-INIT/en/pdf`, accessed on 2024-10-17. 2016 (cit. on pp. 2, 6).

[10] California State Legislature. *California Consumer Privacy Act of 2018.* `https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375`, accessed on 2024-10-17. 2018 (cit. on pp. 2, 6).

[11] Brazilian President of the Republic. *Lei Geral de Proteção de Dados Pessoais.* `http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/L13709compilado.htm`, accessed on 2024-10-17. 2018 (cit. on pp. 2, 6).

[12] Chris Mills. *Saying goodbye to third-party cookies in 2024.* `https://developer.mozilla.org/en-US/blog/goodbye-third-party-cookies/`, accessed on 2024-10-17. 2023 (cit. on p. 2).

[13] John Wilander. *Privacy Preserving Ad Click Attribution For the Web.* `https://webkit.org/blog/8943/privacy-preserving-ad-click-attribution-for-the-web/`, accessed on 2024-10-17. 2019 (cit. on p. 2).

[14] *Preparing for the end of third-party cookies.* `https://developers.google.com/privacy-sandbox/blog/cookie-countdown-2023oct`, accessed on 2024-10-17. 2023 (cit. on pp. 2, 8).

[15] *Preparing for the end of third-party cookies.* `https://privacysandbox.com/news/update-on-the-plan-for-phase-out-of-third-party-cookies-on-chrome/`, accessed on 2024-10-17. 2024 (cit. on p. 2).

[16] Anthony Chavez. *A new path for Privacy Sandbox on the web.* `https://privacysandbox.com/news/privacy-sandbox-update/`, accessed on 2024-10-17. 2024 (cit. on pp. 2, 8).

[17] Eric Rescorla and Martin Thomson. «Technical comments on FLoC privacy». In: *Mozilla, June* 17 (2021) (cit. on pp. 3, 10).

[18] Alex Berke and Dan Calacci. «Privacy Limitations of Interest-based Advertising on The Web: A Post-mortem Empirical Analysis of Google's FLoC». In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security.* CCS '22. Los Angeles, CA, USA: Association for Computing Machinery, 2022, pp. 337–349. ISBN: 9781450394505. DOI: `10.1145/3548606.3560626`. URL: `https://doi.org/10.1145/3548606.3560626` (cit. on pp. 3, 10).

[19] Florian Turati. «Analysing and exploiting Google's FLoC advertising proposal». en. Master Thesis. Zurich: ETH Zurich, 2022. DOI: `10.3929/ethz-b-000539945` (cit. on pp. 3, 10).

[20] Nikhil Jha, Martino Trevisan, Emilio Leonardi, and Marco Mellia. «On the Robustness of Topics API to a Re-Identification Attack». In: *Proceedings on Privacy Enhancing Technologies.* Vol. 2023. Privacy Enhancing Technologies Symposium, 2023, pp. 66–78 (cit. on pp. 3, 15).

[21] *The Privacy Sandbox Timeline.* `https://privacysandbox.com/intl/en_us/open-web/#the-privacy-sandbox-timeline`, accessed on 2024-10-17. 2024 (cit. on pp. 3, 10).

[22] *Preparing to ship the Privacy Sandbox relevance and measurement APIs.* `https://developers.google.com/privacy-sandbox/blog/shipping-privacy-sandbox`, accessed on 2024-10-17. 2023 (cit. on p. 3).

[23] Nikhil Jha, Martino Trevisan, Luca Vassio, and Marco Mellia. «The Internet with Privacy Policies: Measuring The Web Upon Consent». In: *ACM Trans. Web* 16.3 (Sept. 2022). ISSN: 1559-1131. DOI: `10.1145/3555352`. URL: `https://doi.org/10.1145/3555352` (cit. on pp. 3, 23, 25, 33, 45).

[24] Jonathan R Mayer and John C Mitchell. «Third-party web tracking: Policy and technology». In: *2012 IEEE symposium on security and privacy.* IEEE. 2012, pp. 413–427 (cit. on p. 6).

[25] Hassan Metwalley, Stefano Traverso, Marco Mellia, Stanislav Miskovic, and Mario Baldi. «The online tracking horde: a view from passive measurements». In: *International Workshop on Traffic Monitoring and Analysis.* Springer. 2015, pp. 111–125 (cit. on p. 6).

[26] Nikhil Jha, Martino Trevisan, Marco Mellia, Rodrigo Irarrazaval, and Daniel Fernandez. «I Refuse if You Let Me: Studying User Behavior with Privacy Banners at Scale». In: *2023 7th Network Traffic Measurement and Analysis Conference (TMA).* 2023, pp. 1–9. DOI: `10.23919/TMA58422.2023.10198936` (cit. on p. 8).

[27] *OneTrust.* `https://www.onetrust.com/`, accessed on 2024-10-17. 2024 (cit. on p. 8).

[28] *HubSpot.* `https://www.hubspot.com/`, accessed on 2024-10-17. 2024 (cit. on p. 8).

[29] *CookieBot.* `https://www.cookiebot.com/`, accessed on 2024-10-17. 2024 (cit. on p. 8).

[30] *Privacy-Preserving Attribution.* `https://support.mozilla.org/en-US/kb/privacy-preserving-attribution`, accessed on 2024-10-17. 2024 (cit. on p. 8).

[31] *Safari & Privacy.* `https://www.apple.com/legal/privacy/data/en/safari/`, accessed on 2024-10-17. 2024 (cit. on p. 8).

[32] Valentino Rizzo, Stefano Traverso, and Marco Mellia. «Unveiling web fingerprinting in the wild via code mining and machine learning». In: *Proceedings on Privacy Enhancing Technologies* 2021.1 (2021), pp. 43–63 (cit. on p. 8).

[33] Emmanouil Papadogiannakis, Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. «User Tracking in the Post-Cookie Era: How Websites Bypass GDPR Consent to Track Users». In: *Proceedings of the Web Conference 2021.* New York, NY, USA: Association for Computing Machinery, 2021, pp. 2130–2141. ISBN: 9781450383127. URL: `https://doi.org/10.1145/3442381.3450056` (cit. on p. 8).

[34] *Topics API.* `https://developers.google.com/privacy-sandbox/relevance/topics`, accessed on 2024-10-17. 2024 (cit. on p. 12).

[35] Yao Xiao and Josh Karlin. *Topics API: Unofficial proposal draft.* `https://patcg-individual-drafts.github.io/topics/`, accessed on 2024-10-17. 2024 (cit. on p. 12).

[36] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina N. Toutanova. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». In: 2018. URL: `https://arxiv.org/abs/1810.04805` (cit. on p. 13).

[37] Yohan Beugin and Patrick McDaniel. «Interest-disclosing Mechanisms for Advertising are Privacy-Exposing (not Preserving)». In: *Proceedings on Privacy Enhancing Technologies.* Vol. 2024. Privacy Enhancing Technologies Symposium, 2024, pp. 41–57 (cit. on pp. 13, 15).

[38] *Topics API: Model Execution Demo.* `https://colab.research.google.com/drive/1hIVoz8bRCTpllYvads51MV7YS3zi3prn`, accessed on 2024-10-17 (cit. on p. 13).

[39] CJ Carey et al. «Measuring Re-identification Risk». In: *Proc. ACM Manag. Data* 1.2 (June 2023). DOI: `10.1145/3589294`. URL: `https://doi.org/10.1145/3589294` (cit. on p. 15).

[40] Mário S. Alvim, Natasha Fernandes, Annabelle McIver, and Gabriel H. Nunes. «A Quantitative Information Flow Analysis of the Topics API». In: WPES '23. <conf-loc>, <city>Copenhagen</city>, <country>Denmark</country>, </conf-loc>: Association for Computing Machinery, 2023, pp. 123–127. ISBN: 9798400702358. DOI: `10.1145/3603216.3624959`. URL: `https://doi.org/10.1145/3603216.3624959` (cit. on p. 15).

[41] *Browsing context - MDN Web Docs Glossary: Definitions of Web-related terms | MDN.* `https://developer.mozilla.org/en-US/docs/Glossary/Browsing_context`, accessed on 2024-10-17. 2024 (cit. on pp. 17, 41).

[42] *Protocol Buffers Documentation.* `https://protobuf.dev/`, accessed on 2024-10-17 (cit. on pp. 18, 30).

[43] *The Privacy Sandbox enrollment attestation model.* `https://github.com/privacysandbox/attestation`, accessed on 2024-10-17. 2024 (cit. on p. 18).

[44] *sqlite3 — DB-API 2.0 interface for SQLite databases.* `https://docs.python.org/3/library/sqlite3.html`, accessed on 2024-10-17 (cit. on p. 26).

[45] *Chromium Source Code, version 122.0.6261.128.* `https://source.chromium.org/chromium/chromium/src/+/refs/tags/122.0.6261.128:`, accessed on 2024-10-17 (cit. on p. 27).

[46] *Get the Code: Checkout, Build, & Run Chromium.* `https://www.chromium.org/developers/how-tos/get-the-code/`, accessed on 2024-10-17 (cit. on p. 29).

[47] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. «Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation». In: *Proceedings of the 26th Annual Network and Distributed System Security Symposium.* NDSS 2019. Feb. 2019. DOI: `10.14722/ndss.2019.23386` (cit. on p. 33).

[48] *Docker.* `https://www.docker.com/`, accessed on 2024-10-17 (cit. on p. 40).

[49] *Wappalyzer.* `https://www.wappalyzer.com/`, Accessed on 2024-10-17. 2024 (cit. on p. 46).