Bachelor's Thesis Computing Science

Radboud University Nijmegen

An analysis of Javascript's sendBeacon method

Author:
Wout van den Berg
s1014417

First supervisor/assessor:
Gunes Acar

Second supervisor:
Zahra Moti

Second assessor:
Hanna Schraffenberger

April 5, 2024

Abstract

The SendBeacon method was introduced to reliably send client-side analytics data, even when the user closes the page. However, no existing studies investigated its use on real-world websites. In this thesis, we show how JavaScript's `sendBeacon` method is used on the web. Using an extended version of DuckDuckGo's Tracker Radar Collector we perform a web crawl of the 10,000 top-ranked sites according to the Tranco list. In addition to this, we also perform a manual analysis of the `sendBeacon` method. Our research reveals that the `sendBeacon` method is used by 67% of the top-ranked websites. We show that most of the `sendBeacon` calls made on those websites are made by only a few different third-party entities. In addition to that, a manual analysis of the `sendBeacon` method on a few websites reveals that the `sendBeacon` method is sometimes used to send tracking-related data such as pages users visit, contents on those pages, and device information such as screen dimensions to third parties.

Contents

1 Introduction **2**

2 Research **4**
2.1 Extending Tracker Radar Collector 4
2.1.1 Collecting `sendBeacon` calls 5
2.1.2 Capturing `sendBeacon` calls triggered when closing page 5
2.1.3 Testing extended TRC 6
2.2 Analysis 7
2.2.1 Analysis of crawler visits 7
2.2.2 Analysis of automated CMP interactions 8
2.2.3 Analysis of the number of `sendBeacon` calls made . . 9
2.2.4 Sources making `sendBeacon` calls 12
2.2.5 Targets receiving `sendBeacon` calls 13
2.2.6 Analysis of `sendBeacon` arguments usage 15
2.2.7 Manual analysis 16

3 Related Work **21**

4 Discussion **24**

5 Conclusions **26**

6 Appendix **34**

Chapter 1

Introduction

Popular websites frequently make use of different techniques to track or identify their visitors [21, 8]. This can be done for various reasons such as less harmful ones like collecting analytical data or improving security. However, it can also be done for more harmful reasons such as third-party tracking [28] or price discrimination [29]. While researchers are aware of the occurrence of online tracking, laypeople often do not know that they are being tracked or the consequences of online tracking [40]. Examples of methods used to track people are cookie syncing a method used to sync cookies between different websites, browser fingerprinting which can be used to identify visitors based on the unique traits of their devices, and evercookies a countermeasure against users trying to delete their cookies [14, 8]. A problem with these methods is that it is not always clear to visitors that they are being tracked, especially when tracking happens without clear consent [42].

Websites can use HTTP methods to collect analytical data from their visitors, before the `sendBeacon` method other techniques such as the `XMLHttpRequest` method [31] and tracking pixels [38] were used. A disadvantage these other methods have is that web browsers do not reliably send the request when closing the page. Therefore, a newer method, `sendBeacon`, was created [7]. `sendBeacon` works, contrary to XMLHttpRequest, by asynchronously sending HTTP POST requests containing a small amount of data to a web server without expecting a response. The advantages of the `sendBeacon` method are that it does not wait for the response, does not block the unloading of the current page and the requests will be initiated before a page unloads, even when the browser is closed [20]. One specific use case for the `sendBeacon` method is link click analysis. The goal of link click analysis is to keep track of the clicks that a visitor makes on a website, it can be used for first-party web analytics as well as third-party cross-site tracking [46]. Hence, `sendBeacon` can be used to send sensitive information.

The `sendBeacon` method has some limitations. First of all its data size cannot exceed 64kb [25], although this is still quite large for most use cases.

Secondly, the `sendBeacon` method has no callback method, in the case that the `sendBeacon` call does not succeed there is no way for the sender script to handle failures and retry again [47]. What makes the `sendBeacon` method interesting is the recent growth of its usage. When the EU's General Data Protection Regulation (GDPR) came into effect in May 2018 [5] websites had to ask for users' consent to use cookies that track their users. This made tracking users through cookies more difficult for web pages and web pages moved to alternative tracking methods such as fingerprinting [41]. At the same time, the `sendBeacon` method had a significant spike from 23.35% of page loads (in Chrome, on March 1, 2018) to 29.53% of page loads (in Chrome, on April 1, 2018). After that, it continued increasing to almost 60% of page loads according to Chrome Platform status [1]. This means that nowadays the larger part of page loads contain the `sendBeacon` method.

However, even though the `sendBeacon` method is widely used on the web, there is not much literature about how the `sendBeacon` method is used. Popular studies about online tracking talk about the different tracking methods, however, they do not mention the `sendBeacon` method [21, 8, 22]. This study aims to bridge this gap by exploring the usage of the `sendBeacon` method on the web, this leads to the research question:

RQ: How is the `sendBeacon` method used by popular websites?

To answer the research question there are the following sub-questions:

- RQ1: Which parties make the `sendBeacon` calls and who receives the calls?

- RQ2: What kind of data is sent by the `sendBeacon` method?

In this thesis, we are going to crawl 10,000 of the most popular websites according to the Tranco list [4], using a modified version of DuckDuckGo's Tracker Radar Collector web crawler [48] so that it also collects `sendBeacon` calls sent when closing webpages [18]. In this thesis, we will discuss the results of the web crawl, which parties use the `sendBeacon` method the most, which parties receive the most calls, and an investigation of some websites to show practical use cases of `sendBeacon` calls.

Chapter 2

Research

To analyze the usage of the `sendBeacon` method on the web we need to visit websites and document how the `sendBeacon` method is used. First of all, we need to decide which websites we are going to visit. We used the Tranco list, which is a research-oriented top sites ranking hardened against manipulation [33]. Our list aggregates the ranks from the lists provided by Majestic, Crux, and Radar from 08 November 2023 to 07 December 2023 (30 days) [4]. From the list that Tranco generated the top 10,000 websites were taken for our web crawl.

The second thing we need to visit the web pages is a web crawler. For this research, we are using DuckDuckGo's Tracker Radar Collector (TRC), which is a modular, multi-threaded, puppeteer-based crawler used to generate third-party request data for the Tracker Radar [18]. The crawler needs to be extended because for our research because TRC does not:

- collect `sendBeacon` calls
- collect calls made that are triggered on page close events

Hence, the TRC needs to be extended for our research which will be further explained in the next few sections.

2.1 Extending Tracker Radar Collector

TRC uses Puppeteer, which uses the Chrome DevTools Protocol (CDP), to visit websites for a web crawl [3]. Using Puppeteer it creates a browser context which is controlled by using commands from the CDP. To divide different tracking tasks TRC uses collectors such as CookieCollector and RequestCollector, which collect cookie and request details, respectively.

2.1.1 Collecting `sendBeacon` calls

To collect `sendBeacon` calls we modified the API call collector, which intercepts and logs specified function calls the browser makes. By default, the API call collector records numerous tracking-related function calls that return details like cookies, screen dimensions, and keyboard events. However, it does not intercept the `sendBeacon` call. For our research, we removed all default function interceptions from the API call collector to avoid cluttering our data and added one for `sendBeacon` calls.

For every `sendBeacon` call that the crawler records we store the following data:

- **source**: URL of the script that makes the `sendBeacon` call.
- **argument**: the arguments that are given to the `sendBeacon` call. The first argument is the URL that will receive the call. The second, optional, argument is the data (which can be an ArrayBuffer, a TypedArray, a DataView, a Blob, a string literal or object, a FormData, or a URLSearchParams object containing the data to send [7]).

Intercepting the `sendBeacon` will make TRC collect `sendBeacon` calls. However, by just changing the breakpoints TRC does not collect all `sendBeacon` calls made. TRC does not collect `sendBeacon` calls that are made when closing a web page because the collectors' data is saved before closing the page. To collect those calls as well we need to make some further adjustments to TRC.

2.1.2 Capturing `sendBeacon` calls triggered when closing page

MDM Web Docs Page on the `sendBeacon` method explains how the `sendBeacon` call is intended for sending data when the user is finished with a page [7]. Therefore it is important to collect `sendBeacon` calls made when a web page is closed. At the end of a page visit, TRC's collectors' data is saved and after that, the page is closed. Because the page is closed after the collectors are saved, TRC misses the `sendBeacon` calls that are triggered when the page is closed.

We adjusted TRC by adding another collector, the `PageCloseCollector`. This collector is executed before all other collectors and its save (`getData`) method closes the page. Using this approach, the original page crawl method of TRC is not changed but the page close event is triggered before collecting the data from other collectors.

Just closing the page has the problem that some methods rely on CDP commands such as the cookieCollector that runs the `netWork.getAllCookies` in CDP. When the page is closed, certain methods will result in errors. For example, the CookieCollector will fail because there is no page to collect cookies from. To simulate the page close event without causing errors

we tried methods including reloading the page and navigating to the URL `about:blank`, which is the browser's default page for an empty tab. Reloading the page does trigger the page close event but in the logs, it will appear that the page is visited twice which is not desired. Navigating to the URL `about:blank` works and triggers the page close events and therefore collects `sendBeacon` calls made when closing a page.

Using navigation to `about:blank` is not the same as closing a tab. However, in both cases, the browser leaves the loaded page which results in triggering closing events such as the `beforeunload`, `unload`, `visibilitychange`, `pagehide` events. In summary, the `PageCloseCollector` is executed before the other collectors and it navigates to `about:blank` to trigger page close events without causing errors in other collectors.

2.1.3 Testing extended TRC

Websites have different purposes for making `sendBeacon` calls, therefore it would not be unreasonable to expect `sendBeacon` calls to be triggered by different events or different timings on a website visit. For example, for analytics data, it would be more logical to send the data at the end of a visit when the page is closed using the `visibilitychange` event than sending the data at the beginning of the page load at the `onload` event. Because many situations can trigger `sendBeacon` calls it is important to test whether our extended version of TRC captures those calls. Therefore, we made test cases for different events.

The Mozilla documentation of the `sendBeacon` call mentions that the most reliable way to make `sendBeacon` calls for analytics or diagnostics when closing the page is the visibilitychange event [7]. However, events such as `unload`, `beforeunload`, and `pagehide` can be used as well to detect when a user closes a webpage. These events are less reliable because, for example, these events do not fire when closing the web browser on a mobile phone through the phone's app manager [7].

event	**description**	**captured**
unload	fired when the document is unloaded	yes
beforeunload	when the document is about to be unloaded	yes
visibilitychange	fired when contents of its tab have become visible or have been hidden	yes
pagehide	when browser hides the current page in the process of presenting a different page	yes

Table 2.1: Events tested

To ensure that the extended TRC captures `sendBeacon` calls triggered by these events we made some test pages, one test page for each event. We performed a web crawl on the test pages using our extended TRC [49]. Each test page contained a small form consisting of one text field, the data of that form is sent through a `sendBeacon` call triggered by an event. For each event, we made a separate page and checked whether our extended TRC captures the `sendBeacon` call made. Because the page has to be hosted on a server for the crawler to access the page we hosted the pages locally using a Django webserver [2]. The events that we tested are shown in table 2.1 and for every event tested the extended version of TRC captures the `sendBeacon` call.

Although all events tested are captured, this does not guarantee that all `sendBeacon` events made during the crawl are captured. Only a few tested events do not imply that TRC fires all possible events and captures the `sendBeacon` calls made by those events. In addition to that, because the test pages were hosted locally, we had to add a small 3000-millisecond timer before triggering the `sendBeacon` call. The reason for this is that a locally hosted page loads too fast for TRC, and the collectors are not able to collect the `sendBeacon` call when it is made that early on the page load [12]. Because pages on the web load slower than locally hosted pages, it is difficult to test how many, if any at all, `sendBeacon` calls are not captured because TRC's collectors need more time to load.

2.2 Analysis

Using the adjusted TRC we crawled the top 10.000 pages of the Tranco list. The crawl was performed on a household internet connection in the Netherlands, run with 8 concurrent crawlers on the 14th and 15th of December in 2023. TRC can automatically handle consent management policies (CMPs), it does this by trying to look for keywords to recognize CMP patterns. For this, TRC has two modes:

- "optIn": TRC tries to optIn all CMPs, by trying to press the accept option for all policies.
- "optOut": TRC tries to optOut all CMPs, by trying to press the the reject option for all policies.

For this research we visited every website twice, once with the optIn mode and once with the optOut mode. In this section, we show an analysis of the results of the crawl.

2.2.1 Analysis of crawler visits

The crawl did not visit every page successfully, in table 2.2 the number of visits made for both consent modes is shown.

Consent mode	Num. of Crawled Pages	Num. of Successfully Loaded Pages	% Successful
optIn	10,000	9,622	96.22
optOut	10,000	9,619	96.19

Table 2.2: Number of total and successful visits per consent mode

The crawler made 9,563 visits where both the optIn and optOut modes successfully visited the page. The crawler did not visit every page successfully, this can be due to various circumstances, such as internet connection issues on either the crawler's side or the website's side. The optIn mode made 59 successful visits where the optOut mode failed and the optOut mode made 56 successful visits where the optIn mode failed. Given the relatively low number of visits where either one of the two consent modes successfully loaded the page and the fact that the difference is not caused by the consent mode itself, we will exclude these visits in the remainder of our research. This approach makes comparing the optIn and optOut modes more convenient because they now consist of identical visits. By applying this method we still have the 9,563 visits to analyze, which means that the crawl has over a 95% success rate.

2.2.2 Analysis of automated CMP interactions

Besides analyzing the crawler's success rate it is also important to analyze how well the crawler handled consent policies for both consent modes. For every CMP pattern the crawler found on a website it logs that pattern and whether it failed or succeeded to accept or reject the CMP for that pattern. We examined every visited page and divided the pages into three categories:

- `CMPs failed`: TRC did find some CMPs patterns on the page, but it did not manage to accept/reject the CMPs correctly.
- `no CMPs found`: TRC did not find any patterns related to CMPs.
- `succeeded`: TRC did find some CMPs patterns and it succeeded in accepting/rejecting those CMPs.

Figure 2.1 shows how the visits are distributed in these categories. As can be seen in the figure TRC succeeded more with the optIn mode than the optOut mode. This can be explained because a lot of websites contain an accept button for CMPs but no reject button. Therefore, for some websites, TRC can accept the CMPs but it is not able to reject the CMPs because the reject option does not exist on the website. Additionally, you can see that the number of visits where CMPs failed is roughly the same for both consent modes. However, when we look at the number of visits where no CMPs were found we can see that the optIn mode has fewer visits than the optOut mode.

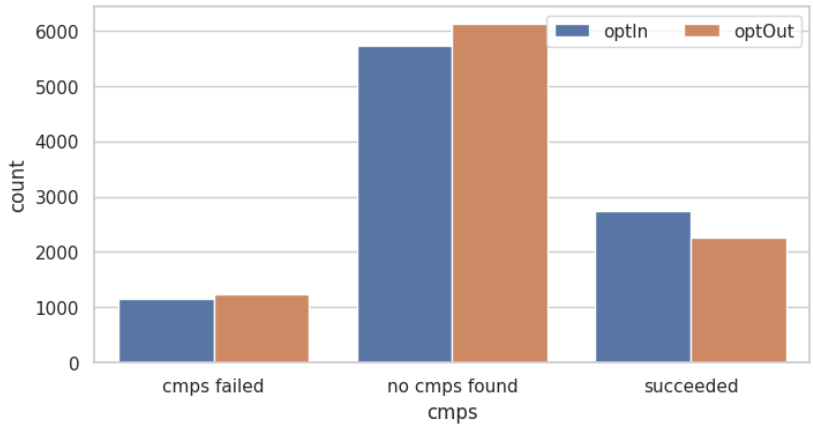

Figure 2.1: CMP results for each consent action

2.2.3 Analysis of the number of `sendBeacon` calls made

Now that the general analyses about the crawl are covered we can examine the `sendBeacon` calls collected by our extended version of TRC. In figure 2.2 the number of visits containing a `sendBeacon` call can be seen.

From the figure, it is clear that the number of visits that made a `sendBeacon` call is higher when the consent mode optIn was used. This indicates that there are at least some websites that change their usage of the `sendBeacon` method based on users' consent. However, it is interesting that even when no explicit consent is given more than half of the websites still make use of the `sendBeacon` method. Figure 2.2 also shows that a little more than 67% of visits contain a `sendBeacon` call for consent mode optIn. This is

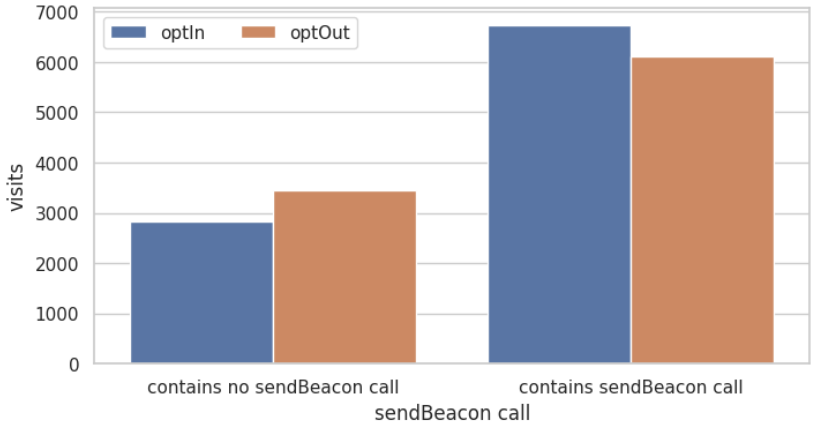

Figure 2.2: Number of visits containing a `sendBeacon` call based on consent mode

higher than the results reported by Chrome Platform Status which reached numbers to up to 58% of page loads containing a `sendBeacon` call [1]. The difference between those two percentages can be explained by taking into account that the crawler only visited websites from the Tranco list whereas Chrome Platform Status is based on all page loads of Chrome users who opted in to share telemetry data with Google.

Figure 2.2 illustrates the number of visits containing a `sendBeacon` call based on consent mode, to give a more detailed picture of the difference between visits containing a `sendBeacon` call we divided those visits into three categories:

1. Visits containing a `sendBeacon` call for both the optIn and optOut modes.
2. Visits containing a `sendBeacon` call for only the optIn mode.
3. Visits containing a `sendBeacon` call only for the optOut mode.

We divided the results of the crawl into these three categories which are plotted in figure 2.3.

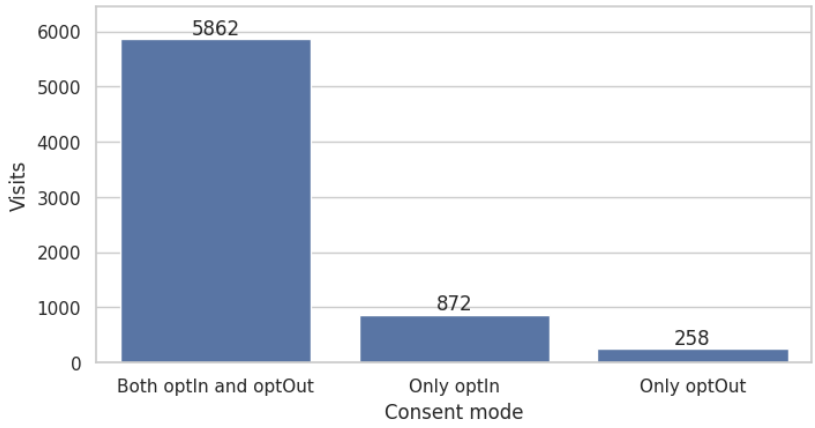

Figure 2.3: Number of `sendBeacon` calls made based on consent mode

As can be seen in figure 2.3 most visits contain a `sendBeacon` call regardless of consent mode. Nevertheless, some visits contain a `sendBeacon` call only for the optIn consent mode, implying that the `sendBeacon` call sends data that requires consent. However, 258 pages made a `sendBeacon` call only when visited with consent mode optOut. This is likely due to noise in the crawls as further explained in the manual analysis in section 2.2.7.

Most websites make multiple `sendBeacon` calls. Websites visited with consent mode optIn made an average of 5.63 `sendBeacon` calls, while those with consent mode optOut made an average of 4.97 `sendBeacon` calls.

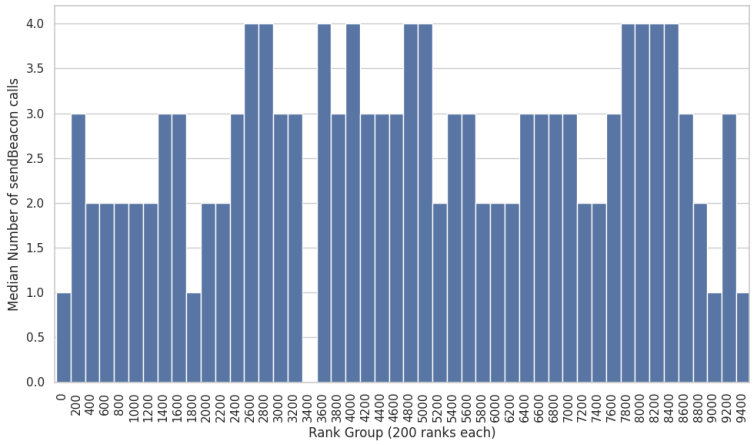

Figure 2.4: Median number of `sendBeacon` calls made with consent mode optIn

Because the average number is sensitive to outliers we also analyzed the median number of `sendBeacon` calls made for each website. We used the Tranco list's ranking and for each 200 ranks, we computed the median number of `sendBeacon` calls made. Figures 2.4 and 2.5 show the median number of `sendBeacon` calls made for each rank group for consent mode optIn and optOut, respectively. The median number of `sendBeacon` calls made is at most 4 for both consent mode optIn and optOut, which is lower than the average number of `sendBeacon` calls. In addition to that, the median number of `sendBeacon` calls that the top 200 ranked domains make is only 1, for both consent modes. Thus, the top-ranked domains make

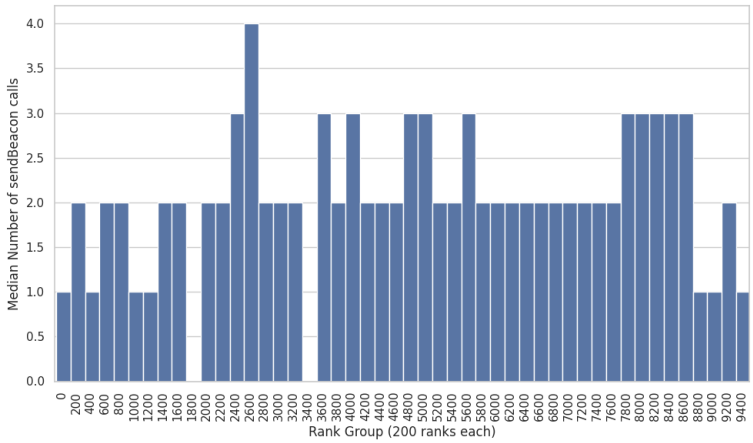

Figure 2.5: Median number of `sendBeacon` calls made with consent mode optOut

fewer `sendBeacon` calls than other domains. A possible explanation for this could be that `sendBeacon` calls are often used by third-party scripts, as mentioned in section 2.2.4, which the top-ranked domains may rely less on. Furthermore, the median number of `sendBeacon` calls made is more constant for consent mode optOut than consent mode optIn.

2.2.4 Sources making `sendBeacon` calls

Often `sendBeacon` calls are made by third parties. In the crawl, 6,992 visits contained a `sendBeacon` call, with 964 containing a `sendBeacon` call made by the first party and 6,583 containing a `sendBeacon` call made by a third party. In this section, we will show which third parties make the most `sendBeacon` calls and the number of `sendBeacon` calls made by those parties. Websites can run external scripts that can make `sendBeacon` calls. For every visit, TRC logs the URL of the script that makes the call and it logs the target URL receiving the `sendBeacon` call.

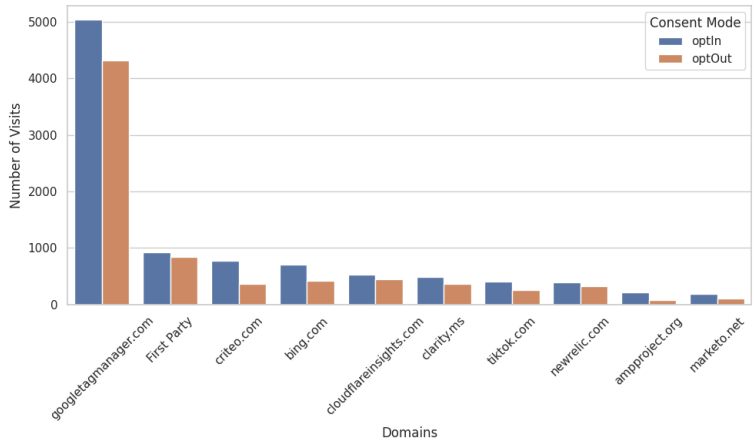

Figure 2.6: Source domains sending `sendBeacon` calls for both consent modes

In figure 2.6 the top ten script domains that call `sendBeacon` are shown, including `sendBeacon` calls made by first parties. The figure shows that the most common source making `sendBeacon` calls is `googletagmanager.com`, and secondly, most `sendBeacon` calls were made by the first party. The figure also shows that different domains behave differently based on consent mode. The domains `googletagmanager.com`, `cloudflareinsights.com`, `newrelic.com`, and the first parties send most of their `sendBeacon` calls regardless of consent mode. Whereas other domains like `criteo.com`, `bing.com` and `ampproject.org` send fewer than half the number of `sendBeacon` calls when no consent is given. This can be partially explained by differences in use case between the domains, some domains use `sendBeacon` calls for

analytics while others use `sendBeacon` calls for tracking-related purposes.

Many companies and organizations operate under multiple domain names, for example in figure 2.6 both `bing.com` and `clarity.ms` belong to Microsoft Corporation. To give a more accurate representation of the number of `sendBeacon` calls made by each entity we grouped the domain names by entity. We did this by using DuckDuckGo's entity domain map [17] to map every domain to an entity name. In the crawl, we found 726 different third-party domains that made `sendBeacon` calls, while when mapped to their entity name 629 different third parties were making `sendBeacon calls`. Figure 2.7 shows the top 10 entities sending `sendBeacon` calls. As can be observed from figure 2.7 the most common source making `sendBeacon` calls is Google LLC. Secondly, most `sendBeacon` calls are made by Microsoft Corporation, unlike figure 2.6, where the first parties were the second most frequent source that made `sendBeacon` calls.

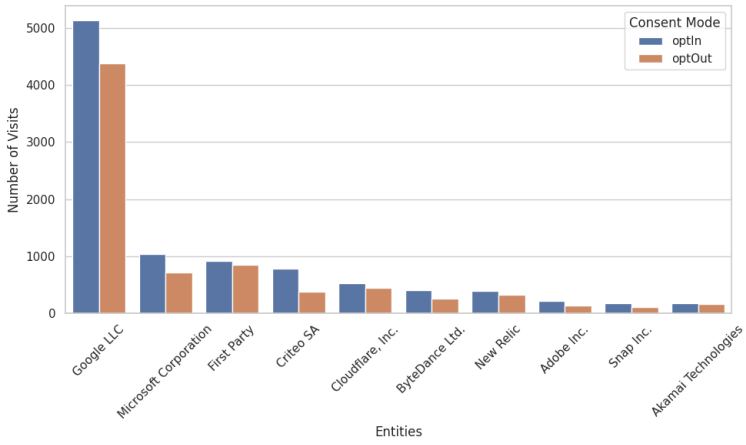

Figure 2.7: Sources sending `sendBeacon` calls grouped by entity name for both consent modes

2.2.5 Targets receiving `sendBeacon` calls

While figures 2.6 and 2.7 show which parties make the most `sendBeacon` calls, it does not give any insights about which parties receive the most `sendBeacon` calls. Figure 2.8 shows which domains received the most `sendBeacon` calls. Note that in this graph, "First Party" means the visited website, not the domain of the script that called the `sendBeacon` method.
Figure 2.8 mirrors the figure 2.6. The domains `criteo.net` and `bing.com` receive fewer `sendBeacon` calls for consent mode optOut, whereas other domains react less intensely to the consent mode given. Google has multiple target domains in the top ten, contrary to the source domains where only one Google domain is included. As in section 2.2.4, we also group the domains

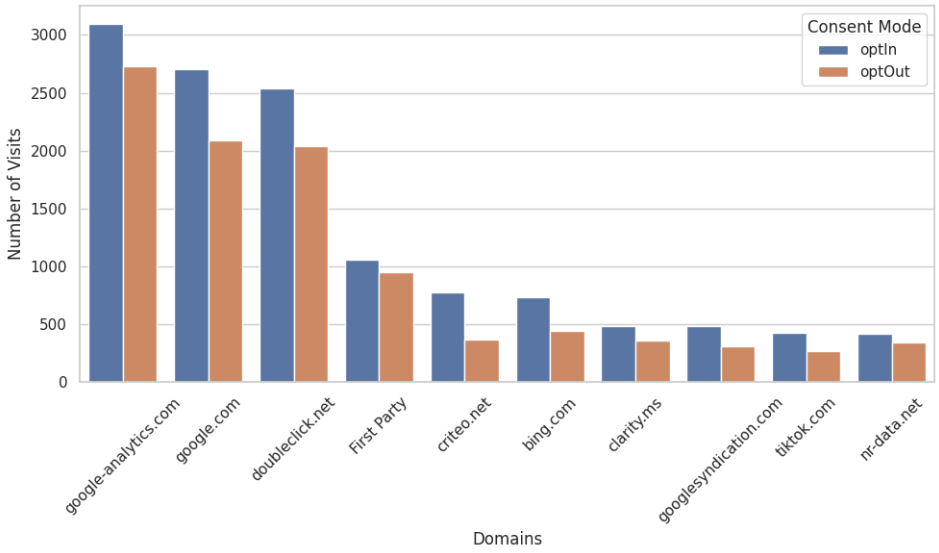

Figure 2.8: Target domains receiving `sendBeacon` calls for both consent modes

based on entity name. In our crawl, we found that 1,139 different third-party domains receive `sendBeacon` calls, and those 1,139 different third-party domains consist of 969 different entities. Figure 2.9 shows the entities that received the most `sendBeacon` calls.

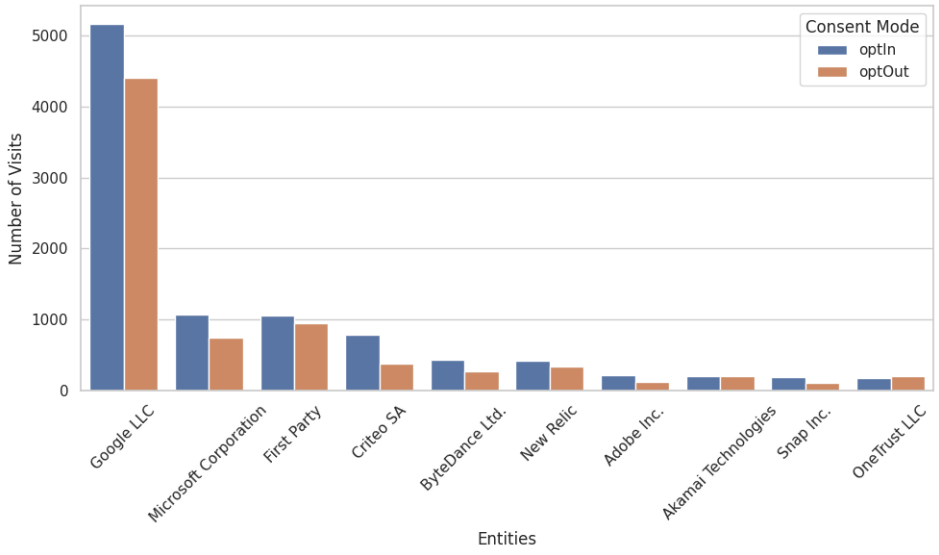

Figure 2.9: Targets receiving `sendBeacon` calls grouped by entity name for both consent modes

As expected, because they send the most `sendBeacon` calls, Google LLC receives the most `sendBeacon` calls. However, something that can be observed from graphs 2.6 and 2.8 is that Google LLC has only one dominant domain sending `sendBeacon` calls while having 4 domains in the top ten receiving `sendBeacon` calls. Another thing that can be observed from figures 2.7 and 2.9 is that while Cloudflare, Inc. sends many `sendBeacon` calls, they do not receive nearly as many calls.

2.2.6 Analysis of `sendBeacon` arguments usage

The `sendBeacon` call has two arguments, the first being the `url` that should receive the call and the second optional argument is `data` that will be sent as the payload. Table 2.3 shows the number of `sendBeacon` calls that make use of the data argument.

The percentage of `sendBeacon` calls containing a data argument is roughly

Consent Mode	**Total num. of SendBeacon Calls**	**With Data Argument**	**Without Data Argument**
optIn	56,326	25,182	31,144 (44.71%)
optOut	49,678	22,599	27,079 (45.49%)

Table 2.3: Number sendBeacon call containing data argument for both consent modes

the same for both consent modes. However, as mentioned in section 2.2.3 most websites made multiple `sendBeacon` calls. Therefore, we show the number of visits containing at least one `sendBeacon` call that uses an argument in table 2.4.

Consent Mode	**Total num. of Visits Containing SendBeacon Call**	**With Data Argument**
optIn	6734	4090 (60.07%)
optOut	6120	3499 (57.17%)

Table 2.4: Number visits containing at least one sendBeacon call with an argument for both consent modes

As can be seen from tables 2.3 and 2.4 the data argument is frequently used. However, besides the data argument, the `url` can be used to send data as well. For example, the URL `https://www.googletagmanager.com/gtag/js?id=G-NCRY038TTP` (57 characters) contains an `id` argument inside the URL parameters. Because URLs can be rather long it is possible to store data inside the URL as

well, for example, the average length of the `url` argument was 86.86 characters and the median length was 56 characters. The `sendBeacon` call with the longest URL length was found on `nationalreview.com` and made by Criteo script. The URL was 9982 characters, whereas the none-parameter part of the URL only contained 46 characters. Therefore, some websites use the URL to send data along with the `sendBeacon` call.

To further illustrate the usage of `sendBeacon` calls we show the object types used. Our extended version of TRC has some limitations regarding object types, TRC captures objects sent through `sendBeacon` calls but does not show the data inside of those objects. During the crawl, we found that 17.43% and 15.14% of visits for optIn and optOut mode respectively contained a `sendBeacon` call with an object.

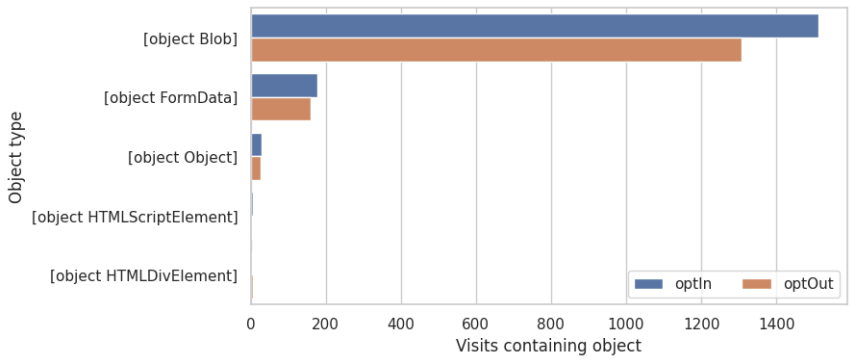

Figure 2.10: Most common objects found in `sendBeacon` calls based on consent mode

In our crawl, 36 different object types were found. A complete list of all object types is shown in table 6.1 in the appendix. Figure 2.10 shows the five most frequently found objects during the crawl. As can be seen in figure 2.10 the object-type `data Blob` is used a lot more than other object types. After that `object FormData` was also used frequently but all object types that are not shown in figure 2.10 are only used by at most 5 websites in total.

2.2.7 Manual analysis

The websites visited with the extended version of TRC reveal information about the usage of the `sendBeacon` calls on a large scale. However, individual use cases of `sendBeacon` calls are not analyzed. In this section, we will show how the `sendBeacon` calls are used on the web by doing a case study on a few websites. For each visit, we scrolled through the page, clicked a few links, and saved the HAR file containing all captured data. The HAR file can be used to examine all requests made by the web page during the visit, the `sendBeacon` calls were identifiable by looking at resource type `ping`.

We repeated this process for both giving consent and not giving consent. This revealed what `sendBeacon` calls were made during the visit, what was inside those calls, and which third-party requests were made for both consent modes.

nyt.com

We visited *nyt.com* and made a short visit of 20 seconds where we scrolled through the website and clicked an internal link. During a visit to *nyt.com* 28 `sendBeacon` calls were made. Most calls were made to *https://a.et.nytimes.com/track* but there were also `sendBeacon` calls made to third-party domains belonging to Akamai Technologies and Criteo. Besides, just the `sendBeacon` calls to third parties more third-party requests were made, all entities that made a request are Datadog, Google, Akamai Technologies, Chartbeat, Brandmetrics, Pickaxe, Criteo, and comScore, Inc.

The `sendBeacon` calls made to a first-party domain of *nyt.com* contained information about all activities during the page visit. The `sendBeacon` calls contained a certain interaction type, for example, `page_update` triggered when going to a different page, `impression` which cause was unclear, and `interaction` when interacting with certain elements on the page. The payload of the `sendBeacon` calls also contained the name of the page, page load time, render time, and the active time of the user visiting the page. During the visit, `sendBeacon` calls were made after events such as clicks or a certain time interval. The `sendBeacon` calls contained information about what events took place, what items were loaded and sometimes even what elements were viewed. Finally, when leaving the page a final `sendBeacon` call was made that included the total time the page was visited.

The `sendBeacon` calls made to Akamai Technologies only contained an 8-digit number inside the URL, and finally the `sendBeacon` calls made by Criteo contained long unreadable URL parameters along with data inside the `sendBeacon` call containing three small numbers.

mediamarkt.nl

During the visit to *mediamarkt.nl* 11 `sendBeacon` calls were made, out of which 10 were made by *googletagmanager.com* going to several domains belonging to Google, and one `sendBeacon` call was made by ContentSquare. During the visit, besides the `sendBeacon` calls several third-party requests were made to Google, Microsoft, Pinterest, Twitter, Oracle, Facebook, ContentSquare, SpeedCurve, User Zoom, The Trade Desk, Criteo, Bazaarvoice, and Cliplister.

The `sendBeacon` call made to ContentSquare contained several query parameters containing a few numbers and a universally unique identifier. In addition to that the request payload contained information about page

elements and their x and y positions.

There were multiple calls made by *googletagmanager.com*, those calls were made towards several different domains belonging to Google LLC. Because *googletagmanager.com* is by far the most frequent initiator of `sendBeacon` calls, we show a bit more about the contents of those `sendBeacon` calls. The target domains and their most important parameters were:

- **googleads.g.doubleclick.net**
  - `url`: the URL of the visited page
  - `gtm`: which is an id of the tag used, which was the same for all `sendBeacon` calls belonging to Google LLC on *mediamarkt.nl*.
- **google-analytics.com**
  - `sr`: the screen dimensions
  - `vp`: the viewport dimensions
  - `dl`: the domain visited
  - `dp`: the domain path visited
  - `dt`: the title of the page
  - `cid`: n UUID
  - `gtm`: gain the id of the tag used (same for all `sendBeacon` calls)
  - `cg1-cg5`: breadcrumbs about the page hierarchy
  - `cd1-cd116`: information about the device, browser, page visited, and more
  - `promo1cr` and similar: URLs to product images shown on the page

As can be seen from the previous lists many tracking-related data is sent through `sendBeacon` calls made by *googletagmanager.com*. When making the same visit again to *mediamarkt.nl* but this time rejecting everything no `sendBeacon` calls were made.

coolblue.nl

To compare the results of *mediamarkt.nl* with another online webshop we also visited *coolblue.nl*. During the visit 12 `sendBeacon` calls were made by Google LLC and 3 `sendBeacon` calls were made by Coolblue. The `sendBeacon` calls that Coolblue made were very short where two only contained information about page load times and one contained information about accepting some Google Analytics tracking cookie notifications.

Besides the `sendBeacon` calls made by Coolblue there most `sendBeacon` calls were made by *googletagmanager.com*. The targeted URLs were again

google-analytics.com and *googleads.g.doubleclick.net*. However, in addition to those two now *region1.analytics.google.com* was also visited, containing again the same `gtm` tag id. In addition to that, it also contained similar fingerprinting-related variables and information about the page visited. Contrary to *mediamarkt.nl*, when rejecting everything there were still similar `sendBeacon` calls as when accepting everything.

Cloudflare

During our sources and targets analysis, we discovered that a substantial amount of `sendBeacon` calls were made by *cloudflareinsights.com* but not as many `sendBeacon` calls were received by *cloudflareinsights.com*. Therefore, we manually visited two websites where we found Cloudflare making `sendBeacon` calls. We visited the domains *brainly.co.id* and *globalsign.com*. During both visits `sendBeacon` calls were made by Google LLC, however, since these `sendBeacon` calls are similar to the ones mentioned in sections 2.2.7 and 2.2.7 we will focus on the Cloudflare `sendBeacon` calls.

Although both domains contained `sendBeacon` calls made by Cloudflare, the targets of those calls were different. For *globalsign.com* the `sendBeacon` calls were made towards *cloudflareinsights.com/cdn-cgi/rum* and for *brainly.co.id* towards *brainly.co.id/cdn-cgi/rum* both without any URL parameters. During both visits, the `sendBeacon` calls made by Cloudflare worked similarly. At first, the `sendBeacon` calls did not contain any payload and occurred at certain events like clicking links on the webpage. However, when triggering the *visibilitychange* event a `sendBeacon` call was made containing payload data. The payload data of both events contained data that described the current and previous window dimensions, the current URL, a `pageloadId`, the referer, and a `siteToken`.

According to some Cloudflare documentation on GitHub: "Beacon data is sent to *https://<yourdomainname>/cdn-cgi/rum* for sites proxied through Cloudflare or *https://cloudflareinsights.com/cdn-cgi/rum* for sites not proxied through Cloudflare. Core Web Vital metrics are reported when the `visibilityState` is hidden for the first time after the page load event is triggered." [16]. Where Core Web Vital Metrics are performance metrics. Therefore target URLs from `sendBeacon` calls made by Cloudflare are still targeted towards Cloudflare-hosted domains. However, it does not appear to be the case that the data sent through the `sendBeacon` calls contain much tracking-related data besides the window dimensions.

Visits making `sendBeacon` calls only when opting out

In figure 2.3 we we showed that 258 crawled websites only contained a `sendBeacon` call for consent mode optOut. In this section, we manually visit a few of those 258 websites to determine whether those cases are noise

or only make `sendBeacon` calls for the optOut mode. Here is a list of websites that only contained a `sendBeacon` call for consent mode optOut in the crawl that we manually visited:

- **newspicks.com** During our visit to *newspicks.com* we did not encounter any consent messages. However, several `sendBeacon` calls were made to different domains of Google LLC and `sendBeacon` calls were made to *bam.nr-data.net* containing some unrecognizable small variables. Looking at the screenshot that TRC made when visiting the page with consent mode optIn it shows that the page was not loaded correctly.

- **oneindia.com** During the visit a message was shown asking for consent. However, even before giving consent several `sendBeacon` calls were made by *googletagmanager.com* containing information about the page visited. For both accepting and rejecting consent the page kept making `sendBeacon` calls. Again the screenshot that TRC made with consent mode optIn showed that the page did not load correctly.

- **fanfox.net** During the visit to the page no consent messages were found, however, some `sendBeacon` calls were made to *region1.google-analytics.com*.

We did not test all 258 visits where only `sendBeacon` calls were made for the optOut mode. However, given the three manually visited websites did not only make `sendBeacon` calls for the optOut mode the 258 visits are probably noise.

Chapter 3

Related Work

Since 2001 numerous studies mentioned the the existence of online tracking [27, 26]. Over time third-party tracking gained more attention, multiple researchers keep acknowledging that it is a growing field [26, 28, 37, 8]. Both stateful and stateless tracking techniques exist, stateful mechanisms require storing a state on the user's device and stateless mechanisms rely on other methods to identify the user [13]. Stateful tracking techniques often rely on cookies that contain some user-identifiable information [27]. Increased awareness of cookies as a means to track users started blocking or removing third-party cookies, therefore countermeasures were invented. An example of such a countermeasure is evercookies, cookies that persist even when the user tries to delete the cookie by storing the cookie in multiple places [8]. Another example is the usage of first-party cookies to set third-party cookies [15].

Contrary to stateful tracking techniques stateless tracking techniques do not rely on storing a state on the user's device but instead use different methods. An example of such a method is the different fingerprinting techniques that allow bypassing cookie restrictions and tracking users across different devices [11, 9]. Despite various online tracking mechanisms that have been researched, research on the usage of the `sendBeacon` method is lacking. Therefore this study could aid in gaining more understanding of the usage of the`sendBeacon` method on the web.

Multiple studies have used a web crawler to study online tracking on a larger scale of the top-ranked websites [21, 8, 19, 15], including studies that modified DuckDuckGo's Tracker Radar Collector to perform a larger web crawl, for example by Senol et al. in 2022 [39]. The paper by Senol et al. mentions the limitations of the crawler against anti-bot measures such as CAPTCHA pages, restricting the crawler from accessing some pages. Multiple studies mention that most third-party scripts belong to a few different entities [26, 21], which mirrors our findings because most `sendBeacon` calls were made by the same few third parties as shown in section 2.2.4. However,

our study differs by focusing on the `sendBeacon` method specifically.

We revealed that the usage of the `sendBeacon` method is higher in the top 10,000 websites than mentioned in Chrome Platform Status [1]. Chrome Platform Status showed a rise in the `sendBeacon` method's usage after the GDPR came into effect, while a 2019 study showed that the use of third-party cookies slightly dropped after the implementation of the GDPR [24]. However, the 2019 study only looked at the top 500 websites, which may give different results than looking at a larger sample size.

Besides crawling web pages, different methods to investigate third-party trackers have also been used, for example, browser extensions [26, 37, 28]. Browser extensions allow researchers to investigate human interaction with web pages, contrary to a web crawl that is automated. Therefore, results could differ when using browser extensions and give more realistic interactions and data captures. A disadvantage of browser extensions compared to web crawlers is that it is not as easy to perform a large-scale study on the web.

In 2016, a 1-million-site measurement and analysis study showed which third parties were most present on the top websites [21]. Comparing their results with our targets and sources found mentioned in sections 2.2.4 and 2.2.5 both include domains from Google. However, other domains differ, our study shows Microsoft, Criteo, and Bytedance as the most common entities besides Google, whereas the 2016 study showed Facebook and Twitter (now x) as the most common entities besides Google. Furthermore, a 2012 study by Roesner et al. showed similar tracker domains as the 2016 web crawl that did also not appear in our top sources or targets making `sendBeacon` calls [37].

Our study only visited front-page URLs from the Tranco list, this has the limitation that the crawler does not visit the same pages as a user would when interacting with a page. A 2020 study by Urban et al. looked at third-party trackers beyond just the front page, showing that only looking at the front page gives a lower bound of the number of third-party trackers [43]. A limitation of this study, however, is that it does not interact with any cookie banners, thus not accounting for differences in consent mode.

A 2023 study by Heino et al. showed that university websites also use third-party services that also face privacy issues, while still applying some methods to prevent personal data from falling into the hands of third parties [23]. This suggests, according to the study, that developers may not always be aware of privacy issues regarding third parties since universities should be exemplary regarding privacy on their websites. However, a limitation of the study is that it did not inspect network traffic and analyze what data the payloads of the requests contain.

While this study focuses on the web, newer studies also investigate the state of third-party tracking in mobile apps [34, 32]. Similar to the web, analyzing mobile apps has difficulties regarding the automated handling of

consent dialogs. Future work could be analyzing whether mobile apps use the same trackers as found in this study to expand the research from web to mobile apps.

Several studies have researched the awareness of the general public about online tracking [40, 44, 45] in addition to gaining informed consent online [30]. The studies show how people lack understanding about the state of online tracking and are unaware of the potential negative impacts of identification and online tracking. A 2019 study revealed that providing information about the state of online tracking to people gave them greater intentions to take privacy-protective actions [45]. Therefore, studies like ours could be relevant since they help to illustrate the current state of online tracking on the web.

In this study, we showed how the `sendBeacon` method is used on the web, however, we did not make a distinction between tracking-related and non-tracking-related `sendBeacon` calls in the results of our web crawl. Several studies mention the difficulty in differentiating between tracking-related purposes and non-tracking-related purposes [15, 21]. This study has the same problem, it is difficult to pinpoint the exact purposes of the `sendBeacon` calls because data is hard to interpret. Different studies have proposed methods to use third-party entities for web analytics without compromising privacy, the studies also show some of the difficulties in guaranteeing that third parties are only used for analytic purposes [10, 35].

Chapter 4

Discussion

There are some limitations to our research. First of all the crawler successfully collects all `sendBeacon` calls that we tested. However, after inspecting the results there are some cases where our extended crawler does not capture all the data. When the `sendBeacon` method sends an object, such as the `object FormData` [6], the values stored inside the object are not saved. Instead of saving the values, the crawler saves the object type as a string without the values, for `object FormData` as "[Object FormData]". This means that the data values that were sent inside the object are lost. As mentioned in section 2.2.6, 17.43% and 15.14% of visits respectively for optIn and optOut mode contain a `sendBeacon` call with an object. The research could be improved by logging the values that are sent inside the objects. Doing this would provide all data that the `sendBeacon` calls sent and hence more information about how the `sendBeacon` calls are used on the web could be researched. The limitation of not collecting object data properly is caused by the usage of CDP by TRC. The TRC binds on JavaScript functions using CDP with Puppeteer. However, since CPD's `Runtime.bindingCalled` method returns a string instead of an object [36] it would require rewriting the TRC beyond the scope of this thesis. Another limitation of our research is that TRC's collectors need time to load, therefore some `sendBeacon` calls could be lost if the page loads faster than the TRC's collectors.

In our research we did not add timestamps to the `sendBeacon` calls, the research could be extended by adding timestamps to the `sendBeacon` calls that are logged. By doing this the timing of the `sendBeacon` calls can be analyzed, it would be useful to analyze whether the `sendBeacon` calls are used at the beginning or end of a visit. Another thing that could extend the research is logging the results of JavaScript methods used on websites that can be used for fingerprinting purposes, such as `window.innerHeight`, and then analyzing whether that data is sent through `sendBeacon` calls. Something that was not looked into in this thesis but could be interesting is looking into the cookies that are set on a webpage and determining whether those

cookies are sent along with the HTTP requests triggered by the `sendBeacon` calls.

Besides the targets that receive the `sendBeacon` calls some CDNs such as Cloudflare and Akamai host other companies' content and scripts. Therefore these CDNs could also receive more `sendBeacon` calls than we found because they receive the `sendBeacon` calls that companies they hosted receive.

The crawl we performed in this study used a desktop Chromium-based browser. Even though the websites that are visited are both accessible from desktop and mobile devices the results could differ. Mobile devices handle certain events differently such as the `unload` and `beforeunload` events [7], therefore the results of this study could differ when the same study is performed on mobile devices.

This study showed the usage of `sendBeacon` calls for the top 10,000 ranked websites on the Tranco list, but it only visited the main pages of those websites. It did not interact with the sites visited like a user would, therefore the behavior of the `sendBeacon` method could differ on some websites that require authentication methods for example. Analyzing this was beyond the scope of this study but could give more insights into how the `sendBeacon` method is used.

As shown in figure 2.3 there were 258 visits where only `sendBeacon` calls were recorded in the optOut mode. However, as mentioned in section 2.2.7 the manual analysis showed that those 258 cases were probably crawler inconsistencies. This brings another limitation to the crawl because there could be more visits for which the `sendBeacon` calls for optIn and optOut mode were not recorded correctly. However, due to the number being relatively low compared to the total number of websites visited it is unlikely to impact the results heavily.

Chapter 5

Conclusions

We used an extended version of DuckDuckGo's Tracker Radar Collector (TRC) to crawl the top 10,000 websites according to the Tranco list. We ran the web crawl using both TRC's optIn and optOut modes, which try to opt in or opt out for all consent policies respectively. Out of the 10,000 attempted visits, 9,563 visits were successful for both consent modes. Analyzing the results of the web crawl we discovered that 67% of visits contain a `sendBeacon` call when trying to opt in for all CMPs. It also revealed a difference in usage of the `sendBeacon` method based on consent mode. However, the difference was only minor because an average of 5.63 and 4.97 `sendBeacon` calls were made for optIn and optOut mode respectively. Observing the median number of `sendBeacon` calls made we discovered that the top 100 ranked websites make fewer `sendBeacon` calls than other websites. Analyzing the sources making `sendBeacon` calls and the targets receiving `sendBeacon` calls we discovered that only a few different parties are responsible for most `sendBeacon` calls, the most common parties being Google, Criteo, and Microsoft.

Analyzing the results of the web crawl showed that usage of the `sendBeacon` method on the web differs based on consent mode. Looking into which third-party scripts make the most `sendBeacon` calls we discovered that only a few different parties are responsible for most `sendBeacon` calls made on the web, the three most common parties being Google, Criteo, and Microsoft.

In addition to the web crawl, we manually analyzed the usage of the `sendBeacon` method. The manual analysis showed that data such as the contents on the page, URL of the page, title of the page, and device information are often being sent using the `sendBeacon` method. This revealed that third parties could be using the `sendBeacon` method to keep track of what pages users are visiting and the content on those pages.

To address the research questions mentioned in the introduction, the `sendBeacon` method is mostly used by a few different third-party entities. In addition to that, the data that is sent through the `sendBeacon` method

is in some cases tracking-related, and in some cases, the data might not be tracking-related. However, regardless of any kind of consent given more than half of the crawled websites make `sendBeacon` calls to third parties that could contain tracking-related data.

Bibliography

[1] Chrome Platform Status. Accessed 2024. Available online: https://chromestatus.com/metrics/feature/timeline/popularity/494.

[2] Django. Accessed 2023. Available online: https://www.djangoproject.com/.

[3] Puppeteer Documentation. Accessed 2024. Available online: https://pptr.dev/.

[4] Tranco List: ID 5Y3WN. Accessed 2024. Available online: https://tranco-list.eu/list/5Y3WN/1000000.

[5] The History of the General Data Protection Regulation, May 2018. Accessed 2024. Available online: https://edps.europa.eu/data-protection/data-protection/legislation/history-general-data-protection-regulation_en.

[6] FormData - Web APIs — MDN, November 2023. Accessed 2024. Available online: https://developer.mozilla.org/en-US/docs/Web/API/FormData.

[7] MDN Web Docs: Navigator.sendBeacon() Method, December 2023. Accessed 2024. Available online: https://developer.mozilla.org/en-US/docs/Web/API/Navigator/sendBeacon.

[8] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, page 674–689, New York, NY, USA, 2014. Association for Computing Machinery.

[9] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. Fpdetective: dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, page 1129–1140, New York, NY, USA, 2013. Association for Computing Machinery.

[10] Istemi Ekin Akkus, Ruichuan Chen, Michaela Hardt, Paul Francis, and Johannes Gehrke. Non-tracking web analytics. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, page 687–698, New York, NY, USA, 2012. Association for Computing Machinery.

[11] Furkan Alaca and P. C. van Oorschot. Device fingerprinting for augmenting web authentication: classification and analysis of methods. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, ACSAC '16, page 289–301, New York, NY, USA, 2016. Association for Computing Machinery.

[12] asumansenol. GitHub - Early browser API accesses and function calls are missed, 2024.

[13] Nataliia Bielova. Web tracking technologies and protection mechanisms. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 2607–2609, New York, NY, USA, 2017. Association for Computing Machinery.

[14] Tomasz Bujlow, Valentín Carela-Español, Josep Solé-Pareta, and Pere Barlet-Ros. A survey on web tracking: Mechanisms, implications, and defenses. *Proceedings of the IEEE*, 105(8):1476–1510, 2017.

[15] Quan Chen, Panagiotis Ilia, Michalis Polychronakis, and Alexandros Kapravelos. Cookie swap party: Abusing first-party cookies for web tracking. In *Proceedings of the Web Conference 2021*, WWW '21, page 2117–2129, New York, NY, USA, 2021. Association for Computing Machinery.

[16] Cloudflare. Understanding web analytics: Data origin and collection, n.d. Accessed on 2024-03-22.

[17] DuckDuckGo. tracker-radar/build-data/generated/domain_map.json at main · duckduckgo/tracker-radar. Accessed 2024. Available online: `https://github.com/duckduckgo/tracker-radar/blob/main/build-data/generated/domain_map.json`.

[18] Duckduckgo. GitHub - duckduckgo/tracker-radar-collector: Modular, multithreaded, puppeteer-based crawler. `https://github.com/duckduckgo/tracker-radar-collector`, 2023.

[19] DuckDuckGo. Duckduckgo tracker radar, Accessed 2024. Available at: `https://spreadprivacy.com/duckduckgo-tracker-radar/`.

[20] Educative. Educative Answers - Trusted Answers to Developer Questions, n.d. Available online: `https://www.educative.io/answers/what-is-sendbeacon-in-javascript`.

[21] 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, page 1388–1401, New York, NY, USA, 2016. Association for Computing Machinery.

[22] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W. Felten. Cookies that give you away: The surveillance implications of web tracking. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, page 289–299, Republic and Canton of Geneva, CHE, 2015. International World Wide Web Conferences Steering Committee.

[23] Timi Heino, Sampsa Rauti, Robin Carlsson, and Ville Leppänen. Third-party services as a privacy threat on university websites. In *Proceedings of the 24th International Conference on Computer Systems and Technologies*, CompSysTech '23, page 134–138, New York, NY, USA, 2023. Association for Computing Machinery.

[24] Xuehui Hu and Nishanth Sastry. Characterising third party cookie usage in the eu after gdpr. In *Proceedings of the 10th ACM Conference on Web Science*, WebSci '19, page 137–141, New York, NY, USA, 2019. Association for Computing Machinery.

[25] Nic Jansma. Strategies for Telemetry Exfiltration (aka Beaconing In Practice), 2020. Accessed 2024. Available online: `https://calendar.perfplanet.com/2020/beaconing-in-practice/#beaconing-payload-limits`.

[26] 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, page 541–550, New York, NY, USA, 2009. Association for Computing Machinery.

[27] David M. Kristol. Http cookies: Standards, privacy, and politics. *ACM Trans. Internet Technol.*, 1(2):151–198, nov 2001.

[28] Jonathan R. Mayer and John C. Mitchell. Third-party web tracking: Policy and technology. In *2012 IEEE Symposium on Security and Privacy*, pages 413–427, May 2012.

[29] Jakub Mikians, László Gyarmati, Vijay Erramilli, and Nikolaos Laoutaris. Detecting price and search discrimination on the internet. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, HotNets-XI, page 79–84, New York, NY, USA, 2012. Association for Computing Machinery.

[30] Lynette I. Millett, Batya Friedman, and Edward Felten. Cookies and web browser design: toward realizing informed consent online. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, page 46–52, New York, NY, USA, 2001. Association for Computing Machinery.

[31] Mozilla Developer Network. Xmlhttprequest, Accessed 2024. `https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest`.

[32] Federica Paci, Jacopo Pizzoli, and Nicola Zannone. A comprehensive study on third-party user tracking in mobile applications. In *Proceedings of the 18th International Conference on Availability, Reliability and Security*, ARES '23, New York, NY, USA, 2023. Association for Computing Machinery.

[33] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. *Tranco list*, 1 2019.

[34] Aidan Polese, Safwat Hassan, and Yuan Tian. Adoption of third-party libraries in mobile apps: a case study on open-source android applications. In *Proceedings of the 9th IEEE/ACM International Conference on Mobile Software Engineering and Systems*, MOBILESoft '22, page 125–135, New York, NY, USA, 2022. Association for Computing Machinery.

[35] Isaac Polinsky, Pubali Datta, Adam Bates, and William Enck. Sciffs: Enabling secure third-party security analytics using serverless computing. In *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, SACMAT '21, page 175–186, New York, NY, USA, 2021. Association for Computing Machinery.

[36] Chrome DevTools Protocol. Chrome DevTools Protocol. Accessed 2024. Available online: `https://chromedevtools.github.io/devtools-protocol/tot/Runtime/#event-bindingCalled`.

[37] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. Detecting and defending against third-party tracking on the web. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, page 12, USA, 2012. USENIX Association.

[38] Jukka Ruohonen and Ville Leppänen. Invisible pixels are dead, long live invisible pixels! In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*, WPES'18, page 28–32, New York, NY, USA, 2018. Association for Computing Machinery.

[39] Asuman Senol, Gunes Acar, Mathias Humbert, and Frederik Zuiderveen Borgesius. Leaky forms: a study of email and password exfiltration before form submission. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1813–1830, 2022.

[40] Fatemeh Shirazi and Melanie Volkamer. What deters jane from preventing identification and tracking on the web? In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, WPES '14, page 107–116, New York, NY, USA, 2014. Association for Computing Machinery.

[41] Alexander Sjösten, Daniel Hedin, and Andrei Sabelfeld. Essentialfp: Exposing the essence of browser fingerprinting, Sep. 2021.

[42] William Tsing. What Does 'Consent to Tracking' Really Mean?, January 2019. Available online: `https://www.malwarebytes.com/blog/news/2019/01/what-does-consent-to-tracking-really-mean`.

[43] Tobias Urban, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. Beyond the front page:measuring third party dynamics in the field. In *Proceedings of The Web Conference 2020*, WWW '20, page 1275–1286, New York, NY, USA, 2020. Association for Computing Machinery.

[44] Jeffrey Warshaw, Nina Taft, and Allison Woodruff. Intuitions, analytics, and killing ants: Inference literacy of high school-educated adults in the US. In *Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*, pages 271–285, Denver, CO, June 2016. USENIX Association.

[45] Ben Weinshel, Miranda Wei, Mainack Mondal, Euirim Choi, Shawn Shan, Claire Dolin, Michelle L. Mazurek, and Blase Ur. Oh, the places you've been! user reactions to longitudinal transparency about third-party web tracking and inferencing. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 149–166, New York, NY, USA, 2019. Association for Computing Machinery.

[46] John Wilander. Link Click Analytics and Privacy, 2019. Accessed 2024. Available online: `https://webkit.org/blog/8821/link-click-analytics-and-privacy/`.

[47] World Wide Web Consortium (W3C). Beacon. Accessed 2024. Available online: `https://w3c.github.io/beacon/#privacy-and-security`.

[48] Woutgit. sendBeacon-crawler, 2024. Crawler used to collect sendBeacon calls. Available online: `https://github.com/woutgit/sendBeacon-crawler`.

[49] Woutgit. sendBeacon-crawler Test Pages, 2024. Test pages for the sendBeacon crawler. Available online: https://github.com/woutgit/sendBeacon-crawler/tree/main/testing.

Chapter 6

Appendix

Object types			
object Blob	object FormData	object Object	object SVGSVGElement
object SVGPathElement	object HTMLSpanElement	object HTMLButtonElement	object SVGPathElement
object HTMLDivElement	object HTMLImageElement	object Text	object HTMLLIElement
object HTMLUListElement	object HTMLHeadingElement	object HTMLParagraphElement	object HTMLIFrameElement
object HTMLInputElement	object HTMLScriptElement	object HTMLLinkElement	object HTMLMetaElement
object HTMLElement	object HTMLStyleElement	object HTMLObjectElement	object HTMLBodyElement
object HTMLHeadElement	object CustomEvent	object Comment	object HTMLFieldSetElement
object HTMLTrackElement	object HTMLOptionElement	object HTMLVideoElement	object DocumentFragment
object HTMLTableElement	object HTMLTableRowElement	object HTMLSourceElement	object HTMLPictureElement

Table 6.1: Object types found in sendBeacon calls