Measuring the Cookie-Setting Behaviour of Web Pages Showing Privacy Warnings

Barnabas Molnar

4th Year Project Report
Computer Science and Management Science
School of Informatics
University of Edinburgh

2020

Abstract

Since the introduction of the GDPR within the European Union, the general web browsing experience for millions of people has changed substantially. A considerable portion of websites have been prompting users to agree to the use of technologies, predominantly cookies, which are aimed to track online behaviour and collect personal data [8]. These prompts are most commonly in the form of privacy dialogs, also known as cookie dialogs. The aim of this study was to develop a research platform, which enables researchers to conduct measurements related to these cookie dialogs. The developed platform, called Cookie Dialog Analysier (CDA), facilitates such measurements to provide an understanding on what effects interactions with these dialogs have in terms of newly placed or removed cookies along with many more useful insights. CDA is intrinsically a web scraper, which was built using Selenium to interact with web browsers [23]. CDA was used to collect cookie dialog related web privacy data from the 500 most popular websites. The findings derived from the gathered data have shown that around 50% of the analysed webpages display privacy notices, and interaction with them generally results in minor shifts in the number of installed cookies. The data analysis generally sheds light on cookie setting behaviour in relation to cookie dialogs and the common wordings and options offered to users through the interface of the notices. The developed system supports the advancement of research around cookie dialogs, which may potentially allow the development and the better enforcement of regulations such as the GDPR.

Acknowledgements

First and foremost, I would like to thank my supervisor, Kami Vaniea, for her relentless support and guidance over the course of my project. Furthermore, I would like to thank my family, my partner and my friends for all their love and encouragement throughout these challenging yet rewarding times.

Table of Contents

Chapter 1

Introduction

Web cookies are indispensable parts of the internet allowing for the implementation of countless crucial and useful functionalities such as logging into an online account or keeping track of a virtual shopping cart. Nevertheless, since cookies also enable the tracking of users' online behaviour, they are capable of completely eradicating online privacy [9]. Web user tracking is facilitated by third-party cookies, usually set by organisations providing ad and web analytics services. Furthermore, these organisations set cookies on the devices of clients who are loading websites or other web-based applications that either belong to them or that have embedded elements originating from them. This hence allows the unique identification of users and the tracking of their online activity and behaviour along with the collection of other sensitive information [16]. It has been shown that a relatively small number of monolithic companies, ubiquitous over an extensive range of websites, have been controlling the collection of personal data [16, 25].

Ever since online privacy has become an issue, policy makers have been trying to regulate the collection of personal data throughout the internet. One of these regulations was designed by the European Union known as the General Data Protection Regulation (GDPR), which became effective on 25 May 2018 [13]. The main aim of the GDPR, regarded as the strictest legal data protection framework to date, is to give control to internet users of the EU over what personal data is being collected about them [39]. Preceding the GDPR, the 2009 revision of the EU ePrivacy Directive has already required websites to request informed consent prior to using tracking technologies [41]. The GDPR extends the ePrivacy Directive by making it stricter through modifications such as demanding more transparent data processing and broadening the definition of personal data. Furthermore, it requires consent to be in the form of an affirmative action, for instance by clicking accept on a consent dialog [20]. After the amendment of the ePrivacy Directive in 2009, websites have started displaying privacy notices, also known as cookie dialogs [41]. These dialogs serve as an attempt to comply with regulations; at the very best asking for consent before using *any* form of profiling technologies such as tracking cookies.

It has been found that despite the GDPR requiring an affirmative action before the use of profiling cookies many websites were still not acting in compliance. Studies conducted after the GDPR was effectuated have shown that around 50% of websites were placing tracking cookies before user consent [14, 41]. Various other studies have analysed the effect of the GDPR and the ePrivacy Directive through examining the usefulness of different forms of cookie consent dialogs and how they affect users [42, 26]. To date, however, there have only been a small number of studies that have investigated the implementation and behaviour of cookie dialogs. Some research has analysed a smaller number of cookie notices manually, while others have selectively focused on groups of dialogs from well-known cookie consent libraries [28, 15, 42, 26]. There has been no in-depth research conducted in this area, however, most likely due to the lack of automated methods for collecting privacy notice data.

The aim of this study was to develop a research platform, allowing the automated and systematic collection of cookie dialogs and related web privacy data with the intention to unravel some of the mysteries surrounding cookie consent dialogs. The goals set out for the platform were for it to be able to locate cookie notices with optional and limited manual assistance. Furthermore, the system was required to interact with the dialogs while scraping essential data that provide insights to the functioning of the dialogs.

In a recent study, Englehardt and Narayanan [16] collected and reviewed a range of web privacy measurement platforms. According to the information provided in the review, these platforms offer different means for collecting website data about online privacy practices measurable from the client side. They allow the detection and the recording of different tracking and profiling techniques, including the collection of cookies. Nevertheless, they are not designed for and hence are not apt to detect and easily interact with specific web page elements such as cookie notices. Due to this shortage, I have designed and built a new platform that allows the automated detection and scraping of cookie dialogs and their features along with other relevant information. Given the great variety of cookie notices, on top of the automated dialog detection tool, I have supplied the platform with a manual annotation tool, which additionally supports the accurate and robust location of dialogs. This platform, called Cookie Dialog Analyser (CDA), uses Selenium, a web browser automation framework to iterate through and analyse 500 of the most popular websites [23]. These websites are retrieved from Tranco, "a research-oriented top sites ranking hardened against manipulation" [27].

I have run CDA on the top-500 webpages from the Tranco list, then analysed the collected dataset with the intention of shedding light on

1. whether cookies are set before cookie dialog interactions,
2. what options are offered by the dialogs, and how prevalent it is for an opt-out choice to be presented by them,
3. what number of cookies, if any, are set as a result of clicking offered options on the dialogs,
4. what portion of cookie dialogs are overlaying websites to the extent that may disrupt or prohibit use
5. and are common wordings used on cookie notices short and concise while also

easily comprehensible.

As websites consists of many highly diverse implementations, the construction of the platform required extensive experimentation and testing to create a system which is fault tolerant and capable of scraping a wide range of webpages. When running CDA on the top-500 webpage list, it was able to locate 71.4% of all cookie dialogs automatically with 92.1% precision. The collected data has been able to adequately answer the aforementioned questions. It has shown that 49.6% of the successfully analysed webpages contained cookie dialogs. CDA has additionally collected a set of primarily first-party cookies before and after dialog interactions. Based on this gathered data 91.4% of the scraped pages contained some type of cookies, and about 20% contained user ID-like cookies.

The collected cookie dialogs contained text with a median length of 57. By ranking the dialog's wordings and their offered options based on their prevalence and significance it was found that the word "cookie" was ubiquitously used, with many of them referring users to privacy policies and settings. The emphasized one-click options offered on the dialogs were often opt-in buttons, with several ambiguous choices for instance for closing the notices themselves. It was further uncovered that clicking *any* of provided options induced the addition of around 1 or less cookies.

To summarise, the three main contributions of this research are as follows:

1. The design and implementation of Cookie Dialog Analyser, a research platform that simplifies the systematic collection of cookie notice and web privacy data.
2. A dataset on cookie dialogs and related web privacy information constructed from a subsection of the most popular webpages globally.
3. Insights on the general implementation and behaviour of a wide variety of cookie dialogs.

The overall structure of the study takes the form of six chapters. Chapter 2 introduces the background necessary to understand several relevant concepts around web privacy and the built cookie-dialog analysis platform. Furthermore, it explores relevant literature by highlighting what pertinent areas of tracking and cookie dialog related web privacy have been studied in the past. The chapter briefly evaluates the methods used to conduct the examined researches and presents their findings. How the platform was built and justifications behind its design choices are presented in Chapter 3. Chapter 4 evaluates the implementation of CDA. The results answering the research questions and their analysis is addressed in Chapter 5. Finally, Chapter 6 gives some final remarks on the outcome of the study, and potential future work is discussed.

Chapter 2

Background

This chapter introduces the background essential for the understanding of this research. The first section familiarises web cookies with the reader. Then, the GDPR and is expanded upon in the context of cookie dialogs and consent. Lastly, literature relevant to the current study are presented and briefly evaluated.

2.1 Web Cookies

Bujlow et al. [7] organized a survey that encompasses various techniques employed for user tracking. They have classified these techniques into five principal groups, namely, session-only, cache-based, storage-based, fingerprinting and other mechanisms. As the list of tracking and user profiling methods is vast, this paper exclusively focuses on the most commonly used method of tracking: cookies. Cookies fall under the category of storage-based tracking mechanisms, which are commonly supported by web browsers and the detection of their use is more straightforward they are explicitly stored on users' machines. Cookies can be installed in a browser in two different ways: they can be set through the HTTP response's “Set-Cookie” header or through making an API call to the server with JavaScript.

Web cookies, invented in 1994, are small files installed on a client's machine by a webserver usually when the client first navigate to it. Cookies are essentially key-value pairs with a small number of attributes. The most relevant attributes for the understanding of this paper are the domain of the website that has set the cookie and the expiration date (specified with the `Expires` and/or `Max-Age` attributes), the time after which the cookie is deleted. Once a cookie, associated with a particular domain d, has been set on a user's machine, the browser may send the cookie at every following request that is made to the server that is linked to d (see Figure 2.1). This allows for numerous key functionalities of the web, allowing websites to have “memory” about visitors.

Cookies have endless possibilities of applications, nonetheless, they can usually be placed into three main categories based on their functions: session management, personalisation and tracking cookies. As the HTTP protocol – primarily used to request

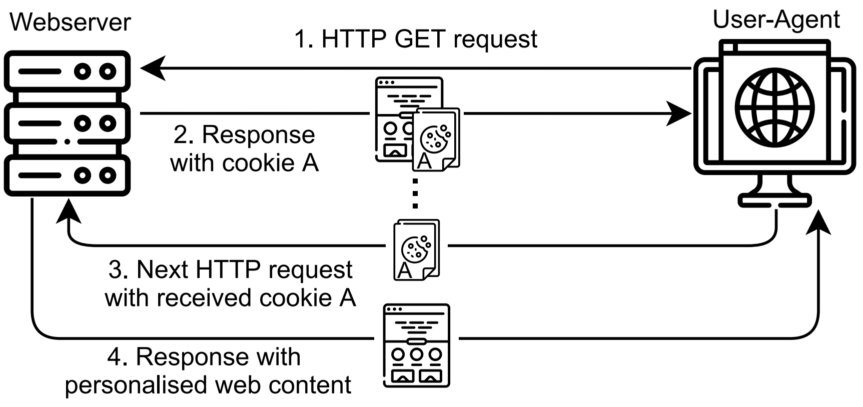

Webserver
User-Agent
1. HTTP GET request
2. Response with cookie A
A
3. Next HTTP request with received cookie A
A
4. Response with personalised web content

Figure 2.1: An example of a general use case of cookies.

web content from servers, is stateless, session cookies were introduced allowing webservers to provide stateful and more personalised functionalities to users . Session management cookies may, for example, enable clients to login to their account or have a shopping cart as they browse. Personalisation cookies permit users to set certain settings according to their preferences such as themes, which can be remembered beyond the end of a browser's session. Finally, tracking cookies, which is a principal focus of privacy research, allow websites to collect and analyse personal information, like browsing patterns and interests of clients [22].

Due to the distinct uses of cookies, different cookies may need to be stored for different periods of time. Session cookie are deleted shortly after a user session ends. However, according to RFC6265, a session is defined by the user agent, that is, the web browser in use [4]. Thus, the meaning of a session cookie is ambiguous across different user agent implementations. Alternatively, persistent cookies may be used which are kept after the end of a session until the expiration date specified in their attributes.

Third-party web elements embedded across a wide range of websites may originate from the same domain, like ads originating from the same ad network. This allows the third parties to install so-called third-party cookies on the user's machine as they browse. When a website requests for a third-party element, the Referer header of the HTTP request is set as the address of the webpage that initiated the request. In this case, the third-party may use the same tracking cookie across all such websites, and from the Referer header it is able to record the user's browsing pattern.

Even if a website requests the loading of a third-party web element, the website cannot read the cookies set by such an element. This is due to an indispensable security mechanism called same-origin policy. This security measure is employed in order to prevent attacks such as Cross-Site Request Forgeries [35].

2.2 GDPR and Cookie Dialogs

The GDPR requires websites to ask for user consent prior to collecting and processing any personal data that is not *necessary* for the basic functioning of the page[1]. This user consent must be in the form of a freely given, unambiguous, affirmative action in response to a clear, concise description of what purposes the data is collected for [13].

In response to the legal requirements of the GDPR, websites have started using cookie dialogs to gain the consent from a user [41]. Nevertheless, it can be argued that numerous cookie dialogs do not live up to the expectations of the GDPR. For example, some dialogs use technical language that is not easily understood by its general audience while others may assume the implicit consent of users visiting their webpage. For examples of cookie dialogs see Appendix A.

2.3 Related Work

There has been extensive research completed in the field of web privacy and web profiling techniques. The subject has been approached from several different perspectives – from a technical perspective, which is crucial to be able to monitor and control the online tracking technologies like cookie use, and form a legal perspective, which is essential for policy makers and regulators. In this section studies in the field of web privacy concerning cookies and cookie notices is reviewed. Most of the examined literatures have a technical stance, however, a limited number of legal papers are considered which are crucial to understand a website's and its cookie dialog's compliance to the GDPR and the ePrivacy Directive.

2.3.1 Web-Tracking-Measurement Studies

A number of studies have been conducted on measuring cookie setting and the use of other profiling techniques around the internet. Most of such studies have conducted such measurements on the most popular websites retrieved from Amazon's Alexa Top Sites ranking, such as [14, 16, 41, 15]. This list could serve as a good representative sample of the typically visited websites of the internet by the general public. This is crucial as most of these researches are interested in user tracking and personal data collection, which largely take place on the sites where majority of users spend their time on the most.

Dabrowski et al. [14] investigated the effect of the GDPR on cookie setting behaviour on the 100,000 most popular websites according to Alexa soon after the GDPR became effective. It explored the differences in cookie setting when the websites were accessed from within the EU, where the GDPR holds, and from outside of the EU, namely from the US. It was found that merely 12.4% of websites accessed restricted its cookie placement in the EU compared to when accessed from the US. Furthermore, the study also examined the changes in cookie setting between pre- and post-GDPR times, i.e.

[1]Several exceptions apply, such as when personal data is collected on the basis of a contract, legal obligations, etc. [13].

between 2016 and 2018. According to the findings, after the enforcement of the GDPR between 30.88% and 46.7% of the top webpages pulled back from installing non-consensual persistent fist-party cookies.

A limitation of this study is that it employs a Google Chrome headless browser to crawl and collect cookies. Such cookies were extracted from the HTTP response headers for both the 2016 and the 2018 scraping. Cookies set by JavaScript, on the other hand, were captured only in the 2018 crawl. As Englehardt and Narayanan [16] observed, crawling with a headless browser (PhantomJS in their case) may result in less accurate measurements as websites may treat a headless browser differently. Webservers may not load sites identically in response to a headless browser as they would in response to a headful browser.

In another study, Trevisan et al. [41] analysed more than 35,000 websites shortly after the of the enforcement of GDPR. The main goal of the research was to discover what proportion of them do not comply with the GDPR and the ePrivacy Directive based on the studied websites' tracking cookie setting behaviours. It was found that 49% of the studied websites are noncompliant with the legislations. A tool called CookieCheck[2] was used to determine whether a website is compliant or not. The tool uses tracker blocking extension lists, in particular, Ghostery[3] and Disconnect[4] to analyse whether a cookie being set is tracking or not. Trevisan et al. [41] also did a manual study on cookie notices and cookie loading after user consent. They uncovered that 28% of websites did not provide a cookie notice at all, and furthermore, of those that do, only 7% retain from setting cookies before consent is given.

Although this gave valuable insight into how prevalent cookie notices are, the study was conducted only few months after the GDPR became effective, which might mean that many websites have not been able to fully implement compliant systems in such short period of time. Furthermore, the paper did not state what options were offered on the studied dialogs and which of those were selected to give consent to personal data processing. In the present study this gap is filled, by associating cookie setting changes with the options that are selected.

Englehardt and Narayanan [16] conducted a comprehensive study in January 2016, before the GDPR was adopted, measuring online tracking on the top-one-million websites, retrieved from Alexa. Their study encompassed not only simple cookie setting behaviour, but also other profiling techniques, such as fingerprinting and cookie syncing. This measurement took a different approach to Dabrowski et al. [14], Degeling et al. [15] and Trevisan et al. [41] since it did not look at tracking mechanisms in a legal context, that is in relation to the GDPR. Instead, it examined different aspects of tracking, such as which third-party trackers are the most pervasive across the studied websites. Englehardt and Narayanan [16] found that throughout the 90 million requests made, 81,000 trackers appeared on a minimum of two websites. However, only 123 of these third parties were present on greater than 1% of the websites analysed. Moreover, many of such third-party domains belong to the same organisation. This

[2]CookieCheck: `http://cookiecheck.polito.it/`
[3]Ghostery: `https://www.ghostery.com/`
[4]Disconnect: `https://disconnect.me/`

justified the observations made in 2009 by Krishnamurthy and Wills [25] that a small number of large companies are controlling personal data collection.

2.3.2 Cookie Warning Notices

Cookie notices have become more and more prevalent after the introduction of the ePrivacy Directive followed by the GDPR. In his article, published not long after GDPR became enforceable, Burgess wrote that 1,051 US based news websites were blocked in the EU. Many websites have started redirecting users, notifying them that their site is unavailable from the EU due to legal reasons. Furthermore, Burgess states that due to the specifications of the GDPR websites have started disrupting the flow of internet use by showing cookie notice pop-ups. According to Brent Mittelstadt, a researcher in Oxford Internet Institute, pop-ups' effectivity will rely on how they are implemented [8].

In another study, Degeling et al. monitored the period when GDPR went effective on 25 May 2018 by scanning on a regular basis the top 500 websites of each EU member country recording any changes relating to privacy policies and cookie consent dialogs. A 16% increase in such dialogs is observed on the studied websites, resulting in 62.1% of websites containing a form of a cookie dialog. Additionally, the research explains through the analysis of consent dialogs of individual websites and consent management libraries that the implementation of the consent requirements outlined by the GDPR is troublesome. The main complications presented include the fact that already installed cookies need to be removed, however, due to the same-origin policy implemented by web browsers, third-party, HttpOnly and Secure cookies cannot be directly controlled by the website itself. In the presence of third-party cookies, a website must rely on APIs for cookie removal provided by the hosts of the third parties that have previously set cookies. Overall, it was concluded that GDPR had a positive influence, creating a more transparent internet, however, it was still missing usable means of opting in and out of personal-data processing [15].

The survey by Degeling et al. was conducted on the impact of the GDPR, which included the analysis of a range of cookie consent libraries. The study nonetheless has not touched upon the technical details of cookie dialogs which do not belong to such libraries, thus leaving it unclear how such notices function other than their ostensible behaviour.

Kulyk et al. [26] conducted a user study on a 150 people measuring the effect of a range of different cookie dialogs on their behaviour. They have concluded that users overall perceive cookie notices negatively. Furthermore, their results have shown that while the text of the dialogs have insignificant impact on whether users leave the webpage, the credibility of the website played a greater role. They have observed that users have little knowledge on what cookies are and what consequences their usage has.

Chapter 3

Cookie Dialog Analyser Design and Implementation

The main objectives of this research was building a web privacy measurement platform, which I named **Cookie Dialog Analyser (CDA)**. CDA is a web scraper at its core which is able to scan through a wide range of websites and interact with privacy notices while extracting relevant information. Building such a tool gives rise to several challenges imposed by the disorderliness of the web and its constantly evolving pool of technologies. To implement CDA, I required a system that is capable of finding privacy notices with adequate precision. The system, furthermore, needed to provide an interface for the manual correction of the located dialogs and the annotation of the dialogs which were not found. Once the dialogs have been accurately located, CDA was required to proceed with the scraping of general privacy and privacy notice related information. CDA had to capture features of privacy dialogs, in addition to pertinent attributes regarding the states of the scraped websites before and after dialog interactions. With such prerequisites, CDA was set to allow the construction of a novel dataset capturing the underlying effects of interactions with generic privacy notices, which are not necessarily tied to cookie consent libraries.

This chapter will describe the realisation of the privacy notice measurement tool, CDA. First, the basic requirements and expectations I had laid out for the system is presented. Second, the design choices with regard to the structure of the system and the chosen technologies for implementation are delineated. Finally, the implementation specifics of the measurement tool are conveyed.

3.1 Measurement Platform Design

Building a platform suitable for the successful extraction and analysis of cookie dialogs is a nontrivial problem. Websites have implementations of such dialogs that use a diverse mix of technologies, such as AJAX, React or Angular, and they may often be constructed in an ad hoc manner, not seldom including broken or obsolete hidden components. To overcome these impediments, I had to make careful design choices.

3.1.1 Basic Requirements and Assumptions

Setting realistic requirements and keeping certain assumptions in mind regarding the measurement platform was vital in order to create a functional and reliable tool that extracts the correct information with minimal human assistance. The basic requirements and assumptions I have formulated with regard to the platform are presented below.

Simulating Real Browsing

A preliminary decision that I had to make was what automation tool to use to allow the simulation of a regular web user's browsing. I prioritised this tool to be suitable to deceive webservers into believing that a human is accessing their website as much as possible. This prerequisite is not rudimentary, as one may assume, since many browser automation tools compromise browser fidelity for speed [16]. As a result, when certain automated browsers make web requests, a non-negligible portion of webservers respond to them with alternative content than they would to a regular browser session's request. Englehardt and Narayanan [16] have conducted a study comparing PhantomJS [37], a headless browser for automated webpage interaction, with OpenWPM, a tool automating a complete browser intended for regular use. According to the results, PhantomJS was observed to load around 30% less HTML. More importantly, they have also found that numerous websites do not provide ads to PhantomJS, likely affecting cookie setting behaviours. Based on these findings, using tools like PhantomJS would thus result in inaccurate measurements leading to conclusions different from what general users would experience.

Fault Tolerance and Robustness

While implementing and testing the system, I have found that because of the wide range of technologies used in webpages, errors are hard to prevent. Thus, I prioritised the fault tolerance and robustness of the system. Possible errors may originate from many sources such unexpected website behaviour due to the websites being A/B tested or using technologies and implementations that were unaccounted for. Consequently, catching runtime errors while ensuring consistency in the output data was an aspect of the program that I needed to keep in mind. Furthermore, to prevent data loss in case the system did fail, regular saves to disk were required. Finally, in case of crashes, I deemed it crucial to implement functionalities that allow the scraping tool to be restarted from the point of the failure.

Extensibility

Another requirement was to make the software easily extensible. This is valuable as it would allow the system to be extended for the use of different experiments in the area of privacy measurements related to cookie dialogs. For example, if a study wanted to measure cookie syncing across domains or whether websites remember past choices regarding tracking, then the system should preferably allow an easy use of the existing infrastructure to embed this new measurement.

Manual Correction of the Detected Privacy Notices

One of the key stages of the scraping procedure is the detection of the privacy warning dialogs. However, the perfect detection of such dialogs is extremely challenging. Thus, a further requirement for CDA is that it provides an interface for the manual correction of the located dialogs and the manual annotation of the dialogs not found during the automated search. Moreover, the use of this tool should be optional subject to what threshold of precision and recall that is required for the dialog detection.

3.1.2 Cookie Dialog Analyser Design: Scraping in Three Rounds

The design of the CDA system, although retaining the base elements, evolved throughout the course of the implementation. The preliminary design envisioned the system to step through a list of websites preforming all tasks on each as one complete and confined process. Its initial design included locating the privacy dialogs, extracting information about the primary state of each website and finally finding and clicking each button on the notices while recording website changes. However, while implementing the system, its complexity increased, and challenges, such as needing to reload after clicking buttons, made it apparent that the process had to be broken down into separate parts. I decided to break the process into three distinct rounds, each of which depend on the output of the round that preceded it. Each round was intended to preform distinct actions and to collect different data:

Round 1 has the aim of locating cookie dialogs on each of the scraped webpages and outputting these. The identified dialogs can then be reviewed and corrected manually before they are used by Round 2. (See Section 3.2.5 for details.)

Round 2 registers the preliminary state of each webpage while also capturing the CSS selectors of the clickable elements within the dialogs and passing them to the next round. (See Section 3.2.6 for details.)

Round 3 iterates through the webpages containing dialogs, clicking each of the links or buttons found in the notices. After each click it collects post-click information about website changes. (See Section 3.2.7 for details.)

As CDA is run, some initial setups are done before any of the scraping rounds are started, including the setup of the automated browser. The implementation and further details of the system are presented in the following section.

3.2 Cookie Dialog Analyser Implementation

CDA consists of three main parts as described in Section 3.1. These are three rounds, each of which have a specific purpose and collect different data. This section will discuss the implementation of CDA, detailing how each of the rounds and their supporting functions operate.

3.2.1 Web Browser Automation

As explained in the requirements of CDA in Section 3.1.1, a web browser automation tool is needed that can simulate "natural" web browsing. I used a popular browser automation tool called Selenium, which is a self-contained system, allowing the automation of popular browsers, such as Chrome and Firefox. Selenium uses WebDriver, which controls standard user agents natively as a web user would. To achieve this, WebDriver offers a set of APIs with which it is possible to fully control web browsers, for example by loading webpages and clicking and manipulating DOM elements in them [43]. It thus became apparent that Selenium meets the requirements as opposed to other user agent automation software, like PhanotmJS. Such automation software are headless, consequently servers may easily detect that a bot is trying to access their web content, even when the user-agent request header is spoofed, as in the experiment conducted by Englehardt and Narayanan [16].

Although numerous web privacy measurement platforms exist such as OpenWPM and FPDetective, as outlined in [16], they do not intrinsically allow easy interaction with webpages. This is because they are specialised for measurements regarding cookies, browser fingerprinting and other specific tracking technologies. On the contrary, the main the goal of this study is to locate and interact with cookie dialogs, thus such technologies are not used for the current purposes. The software however may be extended in the future to collect information about usage of a wide range of tracking technologies, which would make the additional use of such a platform convenient.

Selenium supports the automation of many user agents; however, I came to the conclusion that CDA should use Chrome, the desktop browser that is most widely used as of today [38]. This would allow CDA to generate results that better represent the state of internet privacy. On the other hand, as user agents are not guaranteed to be identical, focusing on Chrome limits CDA from interacting with other web browsers. Nevertheless, a future extension to other user agents is straightforward using Selenium's API.

In terms of implementation, Selenium's WebDriver is set up before the start of each round of scraping. At each start-up, WebDriver starts a Chrome browser instance. This instance is started maximised with disabled browser notifications, which would block any clicks on the main body of a webpage. Furthermore, the browser is started "clean", meaning that it does not contain any information, such as cookies, from previous sessions. The Chrome browser need not only be "clean" at start-up, but also after loading and scraping of each webpage. This is necessary as cookies set at a previous page loads may potentially interfere with new measurements. It might be necessary in future extensions of CDA to preserve previous browser sessions, thus, the restarts can be easily modified by switching a Boolean flag to allow for "dirty" profile browser start-ups.

3.2.2 Analysed Websites

When choosing websites to analyse, it was important to select a subset of the internet that is popular among web users. This allows CDA to construct a dataset that well represents what internet users face on a daily basis. Numerous studies have used Alexa's list of most popular websites [24, 14, 16, 41, 15]. This study deviates from such trends due to two reasons. First, the Alexa top-sites rank only provides a list of the top-50 most popular websites for free, beyond this a payment is required [24]. Secondly, Pochat et al. [32] have shown that Alexa and other popular top-sites rankings may be unreliable as they are easy to manipulate thus possibly altering the outcome of researches. They propose an improved and reliable ranking of websites for the research community based on revised and improved versions of the four most popular website rankings (Alexa, Cisco Umbrella, Majestic and Quantcast). This new ranking is called Tranco, which is used by the CDA system, scanning through the top-500 websites from the Tranco list. I deemed 500 websites to be sufficient for the current study to give a comprehensive insight on cookie notices with the given the computational constraints of my personal computer.

3.2.3 Loading Webpages

Throughout the implementation and testing of CDA many errors occurred because webpages did not load correctly before the system tried interacting with them. These issues occurred for two main reasons. First, in some cases Selenium did not wait for long enough for a webpage to load. CDA handles this by introducing a context manager method, which blocks the program from continuing until the webpage has loaded fully. This wait method uses a heuristic: it employs Selenium's support function `WebDriverWait`, which blocks until the web document's state is ready. Furthermore, it waits an additional 1.5 seconds in case any web elements are loaded asynchronously, such as with AJAX. It is important that a page fully loaded, because cookie dialogs can also be loaded via AJAX.

The second issue arose due to a slight shortcoming of Tranco, which is that their lists do not include the protocol by which the website can be reached together with the domain names they provide (e.g. `yahoo.com` would appear instead of `https://yahoo.com`). This was not an issue for most websites, as their webservers handled protocol changes from `http` to `https` well, however, in some cases the website would not load at all. The webpage loader method detects such errors through looking at both the HTTP status codes and Chrome's error pages (located at `chrome-error://`). If a webpage did not load, then CDA recorded this together with which Round this problem occurred in.

3.2.4 Fault Tolerance

To allow for easy and predominantly supervision-free data collection, it was important that CDA was stable. The internet consists of an extensive range of web technologies, which may cause issues as described in Section 3.1.1 Basic Requirements and Assumptions. Moreover, an unstable internet connection may cause CDA to be interrupted intermittently. Thus, throughout the implementation of the platform several fault tolerance measures were needed. To achieve this, the system has numerous processes wrapped with exception handling blocks to handle faults, such as rare edge cases which were unaccounted for. A large part of implementation consisted of a rigorous experimentation, testing, and error fixing.

Although I have included exception handling in parts of the code that interacted with webpages, program faults may still occur due to various factors, such as bugs in CDA and Selenium or internet connection problems. Thus, I have implemented regular saves after the collection of each batch of data for a website. This allows CDA to be restarted from the point of failure, without the need to rerun any scraping.

3.2.5 Round 1: Finding the Cookie Dialogs

The main purpose of Round 1 is to locate privacy notices on webpages. To achieve this, I had to keep in mind that the perfect automated detection of dialogs may not be feasible because dialogs are generally implemented in very different ways and their structures are often complicated. Recognising this constraint, I had to make a trade-off between recall and precision. I realised that in order to create a tool that could potentially be used to analyse dialogs without the need of manual correction, higher precision would be preferred. If recall was higher, then the results might contain numerous located web elements which are not truly cookie notices. This would then require the manual correction of dialogs even in cases when purely a subset of pages that do contain dialogs is needed.

In light of these decisions, I had to make a choice between three main ideas to detect cookie dialogs:

1. One of the first ideas was to look for common wordings that often occur in cookie dialogs (e.g. “privacy” or “cookie”). On second thought, however, this would have been an unreliable approach, as websites may often contain privacy related phrases elsewhere on the page. Furthermore, due to the potentially complex structure of the dialogs, it would have been difficult to define which ancestor HTML tag corresponds to the whole dialog. This would make the collection of clickable dialog elements imprecise.

2. After the inspection of the source code of different browser extensions that hide cookie banners, such as *Global Consent Manager*[1] and *I Don’t Care About Cookies*[2], I have found that they use static lists which itemise CSS selectors pointing to cookie dialogs. Although these extensions were popular, trying them out lead me to the conclusion that the fact that these lists were static made them become obsolete relatively fast as websites are updated frequently.

3. The idea of using a predefined CSS selector list provides a good path towards locating cookie notices assuming that it is kept updated. Consequently, the solution I discovered and ended up using was parsing a community-maintained filter list called *EasyList Cookie List* [18], which is used as an extension for popular ad blockers. This filter list is used with ad blockers to hide cookie dialogs; thus, it contains an extensive list of CSS selectors. Since it is community maintained,

[1]Global Consent Manager:
`https://addons.mozilla.org/en-US/firefox/addon/global-consent-manager/`
[2]I Don’t Care About Cookies:
`https://chrome.google.com/webstore/detail/i-dont-care-about-cookies/fihnjjcciajhdojfnbdddfaoknhalnja`

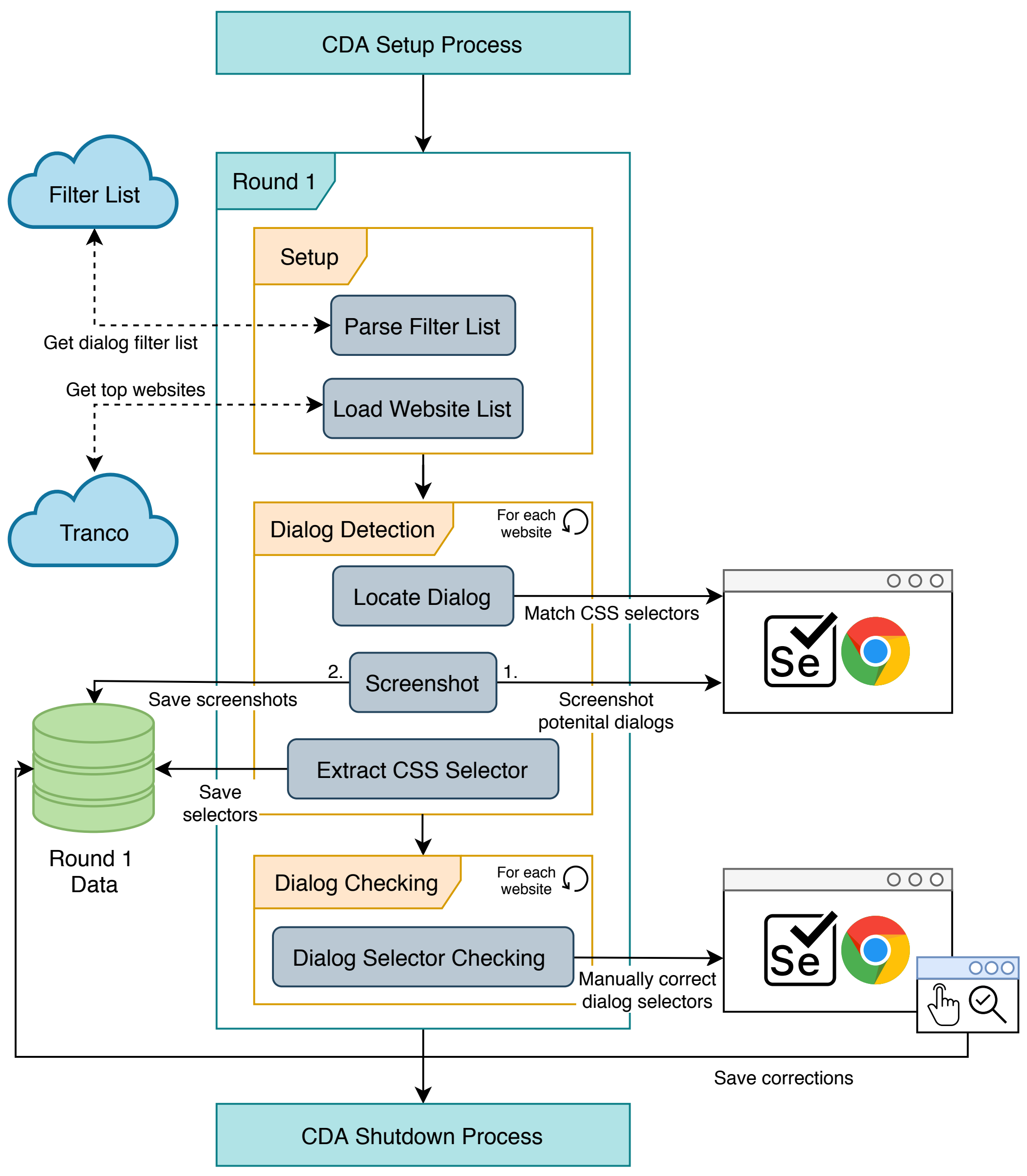


Figure 3.1: Round 1 system diagram.

it is updated daily making it more reliable than the lists within the extensions' source code. Section 3.2.5 will detail the form of the list and how it is used to locate the cookie notices. The choice of using a filter list made the precise location of cookie dialogs possible. This is because the list is specifically crafted for locating cookie dialogs, while other heuristic methods may result in numerous true negative errors, which is undesirable.

The architecture of Round 1 is presented in Figure 3.1. After the setup of CDA, one of the data collection Rounds start. CDA is a single self-contained program, which has to be started with a command line argument (`--round = [round number]`) that specifies which Round to start. When Round 1 is started, the dialog filter list and the website list are obtained from their corresponding servers.

Locating Cookie Dialogs

Detecting cookie dialogs required meticulous testing through experimentation in order to discover and solve problems imposed by the wide range of websites scraped. In advance of locating the privacy notices, the filter list is parsed. This list contains not only CSS selectors, but also different patterns that are used in ad blockers to prevent some requests and scripts from loading cookie notices. The system disregards these instances. The two types of patterns in the list which are used have the following forms:

1. `##[CSS selector]`
   (e.g. `##.notification--GDPR`)

2. `[domain_1,domain_2,...]##[CSS selector]`
   (e.g. `jaguar.co.uk,landrover.co.uk##.NotificationBar`)

In pattern (1.), `CSS selector` is a general CSS selector of any form. If this is matched on any website then it is most likely a cookie dialog. In pattern (2.), `CSS selector` is a website-specific CSS selector, where `[domain_1,domain_2,...]` specifies the list of website domains on which the selector points to privacy notices. The system parses each line of the list using pattern matching with regular expressions defined by the following Python code in order to find the patterns referencing dialog CSS selectors:

Listing 3.1: Patterns for matching wanted filter instances.

```
domain_name_pattern = "([a-z0-9][a-z0-9-]*[a-z0-9]*)"
base_url_pattern = domain_name_pattern + "+(\." +
   domain_name_pattern + "+)+"
list_url_pattern = "((" + base_url_pattern + "),?" + ")+"
```

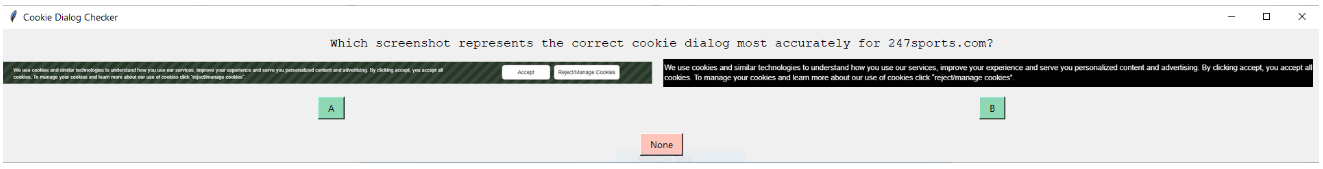
The relevant lines extracted from the lists are then deconstructed and saved for use in the next step. The general selectors are placed into a set while the website-specific selectors are stored in a hash map for efficient lookup.

The following phase consists of the collection of cookie dialogs. The system loads each webpage and obtains all the CSS classes and IDs from them. These are then matched to the selectors derived from the filter list. Initially, if a match was found among the website-specific selectors, then the algorithm did not look at the generic selectors for computational efficiency. In some cases this caused an issue, however, as some of the located dialogs were obsolete and hidden. Thus, to locate the correct dialogs both general and website specific lists are consulted. For each match the filter list's selector is used to query the dialog's Tag object. This was done using BeautifulSoup4, a library for extracting data from HTML files, which allows for faster and better parsing of potentially broken HTML [34]. Through an API call, WebDriver creates a screenshot of each match. Following the screenshots, CDA saves the corresponding selectors to disk.

DialogChecker

After the process of locating each potential dialog, the custom-made DialogChecker is used to manually select which screenshot corresponds to the correct dialog (Figure 3.2). The DialogChecker then marks the CSS selector matching with the screenshot as the unique and accurate selector for the dialog. In short, the DialogChecker is a support tool that makes it easier for a researcher to review the output of Round 1 to ensure that real cookie dialogs have been found and manually mark any dialogs that were not correctly located.

On webpages where no dialog was detected, the user can manually check if there are indeed no dialogs. Furthermore, the selection of the correct dialog is also possible when none of the detected objects are actually cookie dialogs. If a dialog exists despite CDA not having detected it or not having detected the correct object, then this can be rectified through the interface as shown in Figure 3.3a. This is assisted by Selenium, which opens the website with no correctly detected dialogs. This website is opened in Chrome, where an additional extension called *Get Unique CSS Selector* is installed (Figure 3.3b) [1]. The extension allows the user to right-click the cookie dialog and copy its shortest and most exact CSS selector. This selector then can be pasted in the DialogChecker's interface, resulting in DialogChecker updating the website entry's selector.

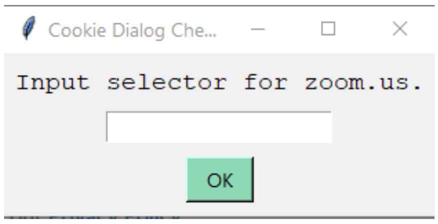


Which screenshot represents the correct cookie dialog most accurately for 247sports.com?

We use cookies and similar technologies to understand how you use our services, improve your experience and serve you personalized content and advertising. By clicking accept, you accept all cookies. To manage your cookies and learn more about our use of cookies click "reject/manage cookies".

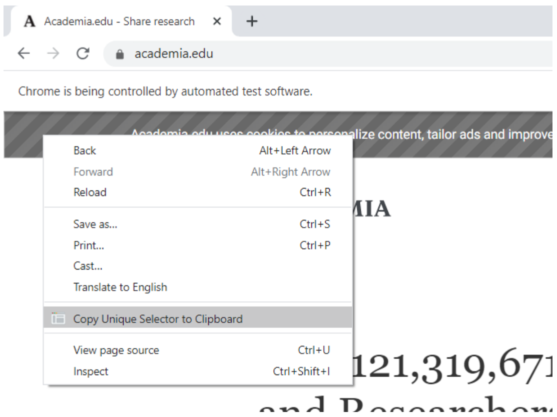
We use cookies and similar technologies to understand how you use our services, improve your experience and serve you personalized content and advertising. By clicking accept, you accept all cookies. To manage your cookies and learn more about our use of cookies click "reject/manage cookies".

Figure 3.2: DialogChecker allowing users to select the most accurate representation of a cookie dialog for a webpage, where two potential dialogs were found. It also allows the user to decide that none of them are correct, in which case an input is provided (see Figure 3.3a)



Input selector for zoom.us.

(a) DialogChecker asking the user to input the selector that points to the correct cookie dialog DOM element.



Academia.edu - Share research
academia.edu
Chrome is being controlled by automated test software.
Academia.edu uses cookies to personalize content, tailor ads and improve
Back Alt+Left Arrow
Forward Alt+Right Arrow
Reload Ctrl+R
Save as... Ctrl+S
Print... Ctrl+P
Cast...
Translate to English
Copy Unique Selector to Clipboard
View page source Ctrl+U
Inspect Ctrl+Shift+I
121,319,671

(b) Copying the unique CSS selector of a cookie dialog is facilitated by *Get Unique CSS Selector* [1].

Figure 3.3: DialogChecker – manual cookie dialog selection.

3.2.6 Round 2: Preliminary Data and Clickable Element Collection

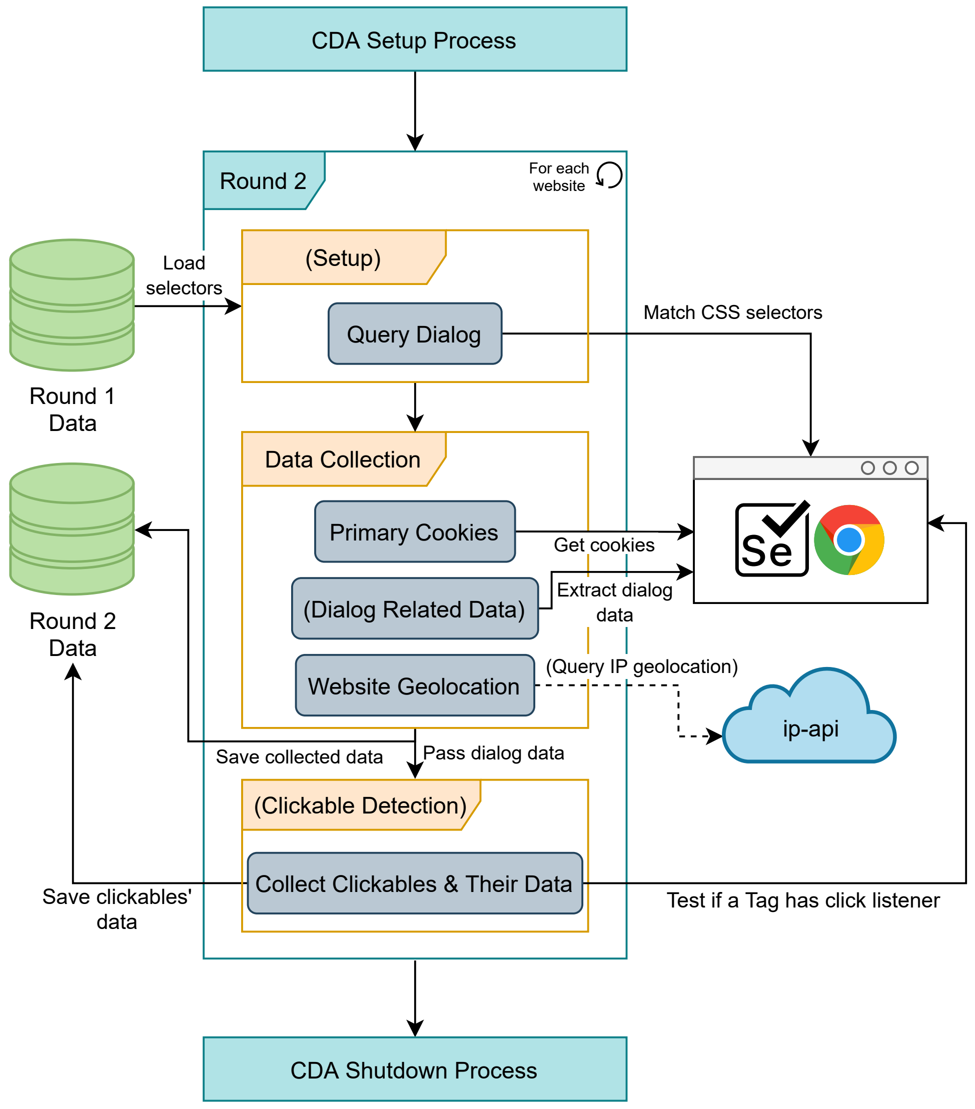


Figure 3.4: Round 2 system diagram.
(abc) – Optional process; depends on circumstances described in Section 3.2.6.

Once the cookie dialogs have been identified on the webpages, the collection of the preliminary data related to the webpages' states and the cookie notice is possible. To initiate this process CDA Round 2 is run with the command line argument `-round=2`. This starts the setup of CDA, including the setup of a Chrome browser instance with the help of WebDriver. Then, the websites are loaded sequentially for each of which the steps described in this section are performed. These steps are visualised in Figure 3.4.

First, given that the loaded page contains a cookie dialog, CDA obtains the required CSS selector from the output created by Round 1 and uses this to query the dialog's Tag. If a dialog exists on the scraped page then as in Round 1, the system uses BeautifulSoup4 to attain the dialog. Finally, CDA collects general information about the website, consisting of the set cookies and the “location” of the website (further details below).

Preliminary Data Collection

The system collects the cookies visible in the browser session of a loaded webpage using WebDriver’s API [31]. Although, the documentation’s wording is imprecise, based on my experience these cookies include mostly first-party cookies, and a limited number of third-party cookies (see Section 3.3). Because this API call queries all cookies that have been set throughout the session, the user agent is restarted, setting up a clean browser between the loading of each webpage. This data sheds light on the number and the type of cookies (e.g. persistent, ID-like) that are set by this specific web page before any potential cookie notice interactions.

Subsequently, the system acquires the “location” of the website, which was primarily meant to identify whether the organisation behind the website is EU based or not. I assumed that this data would give insight on whether EU based websites follow GDPR regulations more. One might believe that a WHOIS request would be suitable for these purposes, which is expected to return the contact details of the organisation or person owning a domain [10]. Nevertheless, as shown by Clayton and Mansfield [10] and Block [6], a significant portion of registrants provide uncomplete information or use privacy services, often provided by registrars to hide personal information from WOHIS records. Thus, WHOIS records are unreliable for identifying the “location” of a website. Hence, I applied a simple approach, where first CDA matches the top-level domain (TLD) of a loaded website to a list of EU based country code TLDs[3]. To do this, the system extracts the TLD of a domain using the Python package *tld* [3]. Secondly, if there was no match, then through an API call to *ip-api.com*, which provides the geolocation of an IP address [2], the continent of the domain’s IP is obtained. This is not a perfect approach as a website may have numerous servers in different continents and may use CDNs as well. However, drawing on the previous findings, WHOIS appears to be less reliable than the current method. As WHOIS potentially provides the domain registrar’s address, it is likely to be more loosely coupled with a website’s organisation’s location than its server location is. The websites where their “locations” were identified by only looking at their TLDs are differentiated in the dataset by assigning them the value `EU`, while the other instances have full continent names spelled out as `Europe` or `Asia`. Overall, this approach gives a good insight on whether an organisation has a target audience in Europe, as only then would they establish domains and servers there.

CDA furthermore collects several attributes of cookie dialogs given that the loaded webpage contains one. These attributes are as follows:

- The text of the dialog, which I believed could be used to give an insight on dialogs’ common wordings and complexity. The text was extracted directly from the dialog’s Tag object returned by BeautifulSoup4.

[3]European Union based country code TLDs: at, be, bg, cy, de, dk, ee, es, eu, fi, fr, gr, hr, hu, it, lt, lu, lv, mt, nl, pl, pt, ro, se, si, sk, uk

- The HTML of the dialog, for future reference and additional analysis. The HTML was obtained by casting the dialog's Tag object to a string.

- Whether the dialog is in the way, blocking users from interacting with the webpage. This was implemented by calculating whether the centre of the dialog is within a 100 pixels of the centre of the window.

- The clickables of the dialog, which aims to include the CSS selectors of all clickable HTML tags within the dialog. The implementation is described below.

Locating Clickable Cookie Dialog Elements

Another crucial step is identifying buttons, links and other web elements with attached click listeners within the cookie dialogs. I refer to these elements as *clickables*. The collection of clickables for each dialog is essential to enable interaction with them in the next round. One of the main difficulties of implementing this step is the fact that essentially any DOM element may be a clickable, and this may not always be obvious from the HTML of a dialog. This is because click listeners may be set with JavaScript on any tag element. Nevertheless, to gain a better understanding on how clickables are most commonly implemented, I inspected a variety of dialogs. I found four prevalent clickable implementations which can be derived from a notice's HTML source code:

1. `<button>` tags,

2. `<a>` tags,

3. `<input type="submit">` tags,

4. and any tag which has the attribute `role="button"`.

When identifying clickables, first, CDA classifies any descendant Tag object of a cookie dialog that falls into one of the categories above. My next goal was to identify other DOM elements which have attached click listeners, however, there appeared to be no native JavaScript methods or other predominant approaches to do so. One option was to only get clickables with listeners where the listener is attached through the HTML attribute `onclick`. This, however, would only find a small subset of such clickables. By digging deeper, I have found that the listeners of a DOM element can be retrieved using the Chrome DevTools console by using the command line API `getEventListeners`. This API, however, cannot be called from an injected JavaScript script using WebDriver. WebDriver fortunately contains an API gateway to Chrome DevTools Protocol, which is primarily used to profile and debug Chrome and Chromium browsers.Thus, the WebDriver method `execute_cdp_cmd(cmd, cmd_args)` provides access to the CDP method `DOMDebugger.getEventListeners`, analogous to the previously mentioned command line API. Finally, the detected clickables are saved for use in Round 3, where they are clicked, and their properties are scraped. The clickables are saved by recording their class, ID and tag name.

3.2.7 Round 3: Cookie Dialog Interactions

Round 3 has the aim to collect data that could shed light on how cookie dialogs respond to different user interactions. It is run by executing CDA with the command line argument `--round=3`. This phase relies on the previous two rounds, where the cookie dialogs and clickables have been gathered. Using this information, CDA can easily locate these privacy notices and interact with them. The process is presented in Figure 3.5. CDA loads each website containing cookie dialogs for each previously collected clickable elements. Once a website is loaded, CDA retrieves the dialog and subsequent clickable object. After CDA obtains these, it collects properties of the clickable and clicks it, recoding site changes. The system restarts the browser between each click, to dispose of any session data, which might interfere with consequent measurements. The following sections explain the implementation of the cookie dialog interactions and the data collection of the clickable properties and the prospective website changes.

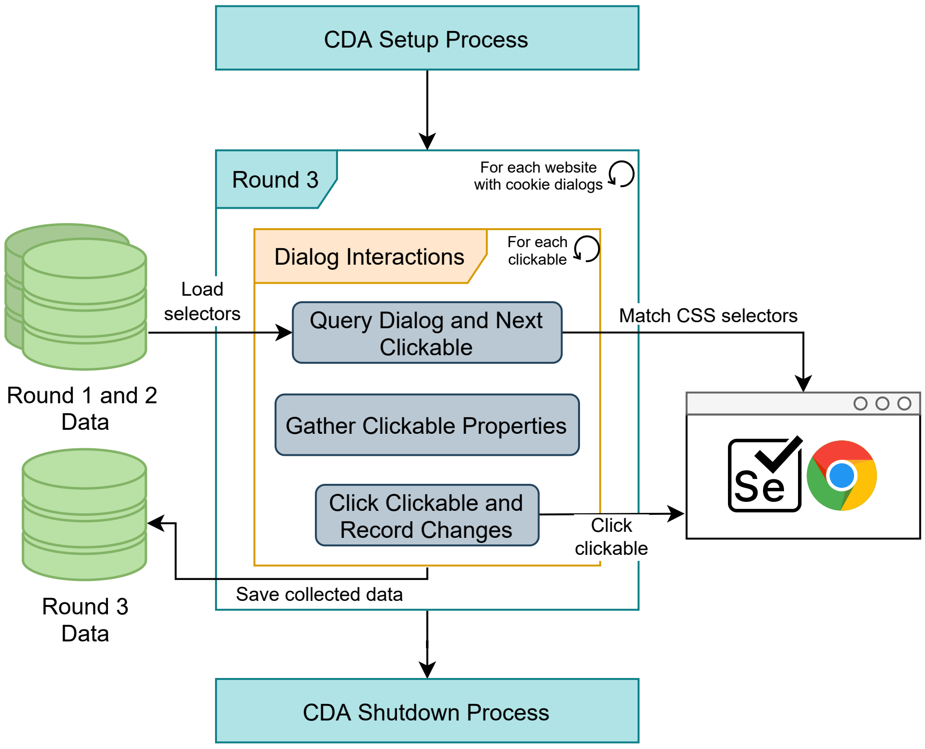


Figure 3.5: Round 3 activity diagram.

Clickables

Before interacting with the privacy notices, their clickable elements are located again using the clickable properties recorded in the previous rounds (the dialog CSS selector and the clickable's IDs, classes and tag name). WebDriver is used to locate them with a constructed CSS selector of the form:

```
[dialog selector]  [tag name][#ID_1#ID_2...][.class_1.class_2...]
```

The dialog selector followed by a space, which is called a descendant selector, imposes the restriction that only clickables within the dialog are selected [11]. This ensures that WebDriver does not select other clickables with the same properties that fall outside the scope of the dialog. Once a clickable is found, CDA records some of its attributes and then clicks it using WebDriver's API if possible. These attributes are the clickable's hyper reference (`href`), text and background colour. Moreover, it is noted whether the clickable is visible or disabled, and they are only clicked if both of these properties are true. Furthermore, in the rare case when another element happens to overlay the clickable then WebDriver throws the exception `ElementClick-InterceptedException`. This behaviour is acceptable as a user would similarly be unable to click the overlaid element.

In general, the text can be easily extracted from a Tag element, however, in some cases clickables contain images representing non-textual symbols. In the context of cookie notices, I believed that it would be useful to understand changes when a dialog is closed, without a user explicitly giving consent to or opting out from being tracked. It is common for such clickables to contain an $\times$ symbol in the form of an image. To be able to capture these cases, I have used *pytesseract*, an optical character recognition (OCR) tool [21]. I expected this tool to translate an input image representing a $\times$ to the character *x*. An issue that I incurred, however, was that the clickable's images were in some cases sprite sheets (see Figure 3.6) [12]. Thus, WebDriver's API to create screenshots of web elements was used to obtain an image of the clickables, which would be input to the OCR tool. A downside of this, however, was that WebDriver creates rather low-resolution screenshots, which impacted OCR's capability to recognise the $\times$ symbol. In instances when pytesseract could not recognise the symbol correctly, I observed that the output string often contained one of the following characters: <, >, `4`, `/`, `\`, and its length was less than 4. Consequently, CDA checks if this is the case, and if so, the clickable's text is recorded as an *x*.

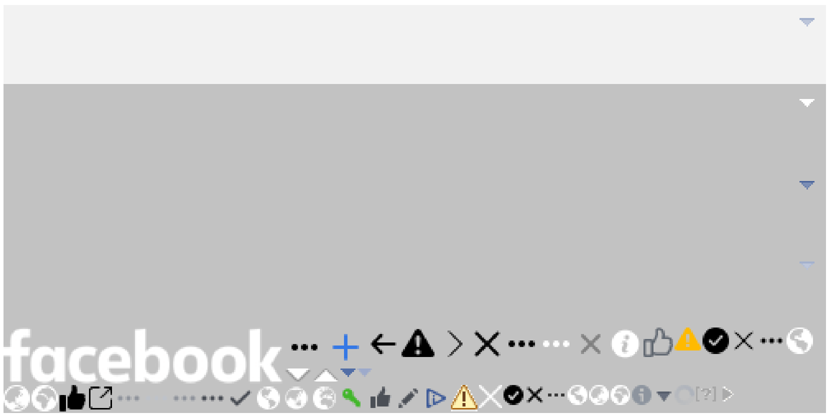


Figure 3.6: A sprite sheet used by one of the clickables, which allows users to close the cookie notice on `facebook.com`. Sprite sheets are used to reduce the number HTTP requests made when loading a webpage [12].

Post-Click Data Scraping

After each successful automated click, CDA aims to record three types of website changes. Primarily, the system registers the session's cookies, which may be used to see if the click has resulted in any new cookies being set. Secondly, CDA records the URL of the webpage after a click, a change in which may suggest that a privacy policy for example was opened. The click may result in a new tab opening, which is managed, by closing the tab navigating back to the original page with its associated window handle. On the original page, CDA records if the click closed the cookie dialog by querying the dialog using its CSS selector.

3.3 Issues Experienced With Retrieving Cookies

Over the course of the year I believed that WebDriver's API, used by CDA, registers all cookies, including all third-party cookies. However, shortly before the project deadline I noticed this was not the case. Surprisingly, the results obtained during the data analysis have shown that a lot smaller portion of the cookies were third party than many related studies have found [15, 16, 14].

A range of factors have misled me to believe that *all* cookies were captured. These factors include that a selection of third-party cookies were indeed recorded. Moreover, the main focus of this project was the collection of dialog related data. In addition, WebDriver has a vague documentation on what cookies are captured by it [31]. Upon further investigation, however, I have found that mostly first-party cookies were captured in reality. WebDriver collects a small subset of third-party cookies as well, which, based on my observations, were set by domains related to the website being visited[4], however, after rigours research no further information was found on what subset this is.

Overall, cookie dialogs have limited control over what cookies third parties install or remove as cross-domain cookie setting is not possible [4]. Thus, I believe that the data obtained by this version of CDA, using WebDriver's API to collect cookies, still gives a valuable insight on cookie setting behaviour in relation to cookie dialog interactions. Changes in the number of recorded cookies in response to a dialog interaction should still describe the behaviour a dialog well. On the other hand, the collection of cookies can be improved upon in a future study where the focus is shifted from the implementation of the dialog scraper.

I have implemented a potential algorithm which is able to collect all cookies, however, it is still a work-in-progress. The method requires CDA to access the *SQLite Cookies* database located under Chrome's User Data directory. There were two main complications, one of which is yet to be resolved.

First, since the release of Chrome 80, the database encrypts all cookie values using a new method. This method first creates an encryption key, which is also encrypted using the Data Protection API (DPAPI) on a Windows system. Furthermore, the cookie values are encrypted with the encryption key with AES-256 in GCM mode. To decrypt

[4]e.g.: On `http://sharepoint.com/` some third-party cookies were captured, which were set by `http://microsoft.com/`.

the cookies, the newer version of CDA first obtains the encrypted key and then decrypts it using `win32crypt.CryptUnprotectData` DPAPI [29]. Subsequently, following the method described in [40], using the decrypted key, CDA decrypts the cookie values.

Second, when CDA tries to access the Cookies database it is *intermittently* empty, even though cookies have been installed and are visible through Chrome's DevTools panel. CDA closes WebDriver to make sure that Chrome writes the required data to disk, however, the database still remains empty. I have found that others have been experiencing the same problem [19], but no solution has been documented to this date as far as I am concerned. This issue is yet to be resolved in a future work as the time constraints have not permitted me to find a solution. Nevertheless, using this technique CDA manages to capture all cookies on a subset of the webpages it iterated through allowing me to capture a sample dataset for evaluation.

Chapter 4

Evaluation

This chapter evaluates the performance of the Cookie Dialog Analyser (CDA) platform in terms of how well it has fulfilled the research requirements. The chapter consists of four sections, the first of which will assess the overall stability of the platform. The remaining three sections will evaluate three aspects of CDA – webpage loading, cookie dialog detection and clickable identification – which are vital for accurate and effortless cookie dialog and web privacy related data collection.

4.1 Fault Tolerance

As described in section 3.2.4, CDA creates regular writes to disk so that it in case of a crash it can be restarted from the point of failure. In general, all data scraping Rounds of CDA run smoothly, however, in some cases bugs or other connection problems may cause interruptions, causing CDA to halt with an error message. All Rounds of CDA were run fully at least once, with multiple experiment runs on different smaller subsets of the top-500 websites. These runs were executed with an unreliable Wi-Fi connection, which implied that in some cases CDA had to be restarted. On average restarting CDA was necessary once for every 375 webpage scrapings in the final full runs among all three Rounds. In case of larger scale scrapes this would be troublesome, however, with a more reliable Ethernet connection the error rate is likely to be considerably lower.

4.2 Loading Web Pages

Across all three Rounds, CDA has been able to fully analyse 88%, that is 444 of the 500 webpages loaded. Round 1 had the greatest number of webpages not loading properly, where 7.4% of the scraped websites did not load during the scraping phase, and 0.2% of websites were unable to load during the dialog checking phase. Furthermore, 3.9% of the webpages successfully scraped in Round 1 were not loaded in Round 2. All remaining websites were successfully analysed in Round 3.

A portion of the webpages not loading were due to HTTP server or client errors, such as timeouts. Some of these issues were due to unresponsive servers or brief network outages. Other websites in the Tranco list which did not load were not meant to be browsed manually by end users, such as `http://cdninstagram.com/`, which is used to deliver image and video contents geographically close to end users. Overall CDA's website loading performed well, however, in the future one improvement that could be made is the detection of short network outages, while CDA would wait until the connection is re-established.

4.3 Locating Privacy Notices

When building CDA, as described in Section 3.1.1 Basic Requirements and Assumptions, the main goal was to locate cookie dialogs with high precision and a low false positive rate. By using a filter list, I have made a trade-off between precision and recall in favour of a higher precision.

Although using a filter list the exactness the detection of cookie dialogs may vary over time, during the final main run of CDA, it has detected 177 cookie dialogs of the 248 present on the loaded webpages (71.4% recall). Of all the detected web elements, 7.9% were erroneously detected (92.1% precision). These erroneously located elements were mostly descendants of its webpage's cookie dialog's root HTML tag. In some other cases, however, mistakenly selected elements were sections of sign-up pages which had opt-in checkboxes for their privacy policies or opt-out parts of digital marketing webpages. These did not qualify as cookie dialogs, so I discarded them using the DialogChecker tool. Overall, CDA preforms reasonably well at detecting cookie dialogs, and the produced dataset with the help of the DialogChecker has likely located all the dialogs on the successfully loaded webpages.

4.4 Locating Clickable Elements

As manually checking and correcting the clickables located by CDA would be eminently tedious, there is no extensive control set to help measure the performance of this step. Nevertheless, I believe the discovered HTML tags and attributes together with the ChromeDevProtocol locating elements with click listeners allows CDA to detect most clickables with good accuracy. After conducting a small-scale measurement on 15 random websites from the top-500 Tranco list, I have found that 97.5% of clickable elements were identified.

Chapter 5

Data Analysis and Results

This chapter presents the insights gained from the scraped data collected with the Cookie Dialog Analyser (CDA). It further outlines some data analysis methods crucial for the interpretation of the presented data. First, in Section 5.1, the cookie setting behaviour of webpages before privacy notice interactions is presented. This section is associated with the data collected in CDA Round 2 (see Section 3.2.6). Then, in Section 5.2, the main focus is cookie dialogs. This section predominantly analyses the data gathered in CDA Round 3 (see Section 3.2.7).

CDA analysed the top-500 websites as of 7 March 2020, retrieved from Tranco[1] [27]. Of these websites, 444 were successfully scraped; thus, all the analysis is established on this subset of websites. Furthermore, unless otherwise stated, the presented results and the conclusions drawn are based on the dataset in which CDA has recorded cookies using the WebDriver API. As described in Section 3.3, these cookies are primarily first party cookies, with some exceptions.

5.1 Initial Cookie Setting Behaviour

This section investigates the initial cookie setting behaviour of the analysed websites. The presented data describes the number and the type of cookies set prior to any clicks made on the cookie dialogs themselves. The term *initial cookie* will be used to refer to such cookies.

Figures 5.1 and 5.2 present the main initial cookie setting statistics. The findings show that of the 444 analysed websites 406, i.e., 91.4% of websites have installed at least one initial cookie. On the first visit of a webpage the number of cookies set had the mean of 9.6 cookies (see Table 5.1 for further summary statistics). These cookies may belong to any of the main cookie categories described in Chapter 2 Background, which consist of session management, personalisation and tracking cookies [22]. The main interests of this study are tracking cookies, which can be linked to users and their personal data collected by the organisations behind websites. It is hard to differentiate such cookies

[1]Available at `https://tranco-list.eu/list/6QPX`.

Dataset	Cookie Type	Mean	Median	First Quartile	Third Quartile	Mode	Max
A	*All*	9.61	7	4	13	5	52
	All (European)	9.42	7	3	13	0, 5	52
	First Party	9.02	6	3	13	0	52
	Third Party	0.60	0	0	0	0	21
B	*All*	18.06	8	0	24	0	211
	All (European)	19.06	8	0	26	0	211
	First Party	7.66	4	0	12	0	58
	Third Party	10.40	3	0	10	0	170

Table 5.1: Summary statistics of the number of cookies set at the first visit of a website, before any cookie notice interactions. Dataset A is the primary dataset, which has used WebDriver's API to collect mainly first-party cookies, and Dataset B is a sample dataset gathered by the new version of CDA collecting all cookies (see Section 3.3).

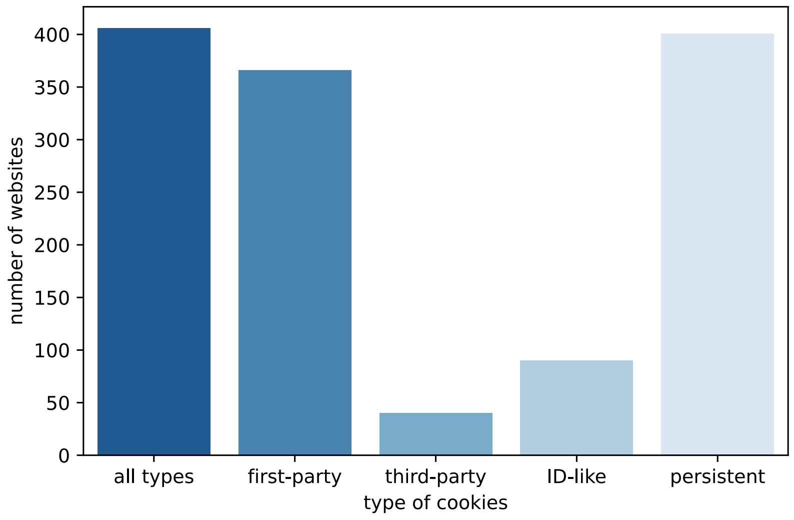


Figure 5.1: Initial cookie setting behaviour: *the number of websites that have set cookies before any cookie notice interaction, which is broken down for first-party, third-party, ID-like and persistent cookies.*

from others which are *necessary* for the basic functioning of a website's features. Unless a cookie is documented, a tracking cookie would only be distinguishable with an insight into the backend webserver. Nevertheless, there are some heuristics for tracking cookie identification used across web privacy measurement studies, such as in [36, 17]. Cookies which are able to track a user over time are persistent, as session cookies are usually deleted after a browser session is closed. Figure 5.1 shows that a majority (90.3%) of websites have set persistent cookies at their first visit.

A portion of third-party cookies are captured by CDA. The distribution of the number of initial third- and first-party cookies set by the visited webpages is presented in more detail in Figure 5.3. It can be observed that in both cases the distributions are skewed, with most websites setting around 8 initial first- or third-party cookies.

To test my hypothesis that organisations hosting websites with a European target au-

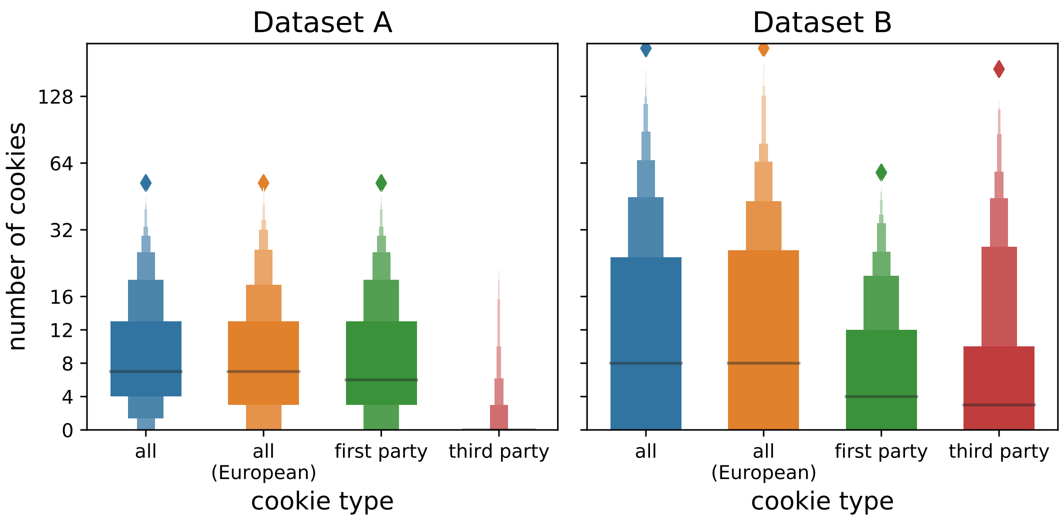


Figure 5.2: A letter-value plot depicting the distribution of the number of initial cookies before any cookie notice interaction. The black horizontal line represents the median value. The largest boxes represent the range from 25^{th} to 75^{th} percentiles. Each following box represent half of the remaining percentiles (e.g.: the following percentile ranges are depicted $[...(12.5,25),(25,75),(75,87.5)...]$. The number of initial cookies is further broken down for websites likely targeting an European audience and first- and third-party cookies. (N.B.: The y axis has a log_2 scale for values $v > 16$ to allow extreme values to fit.)

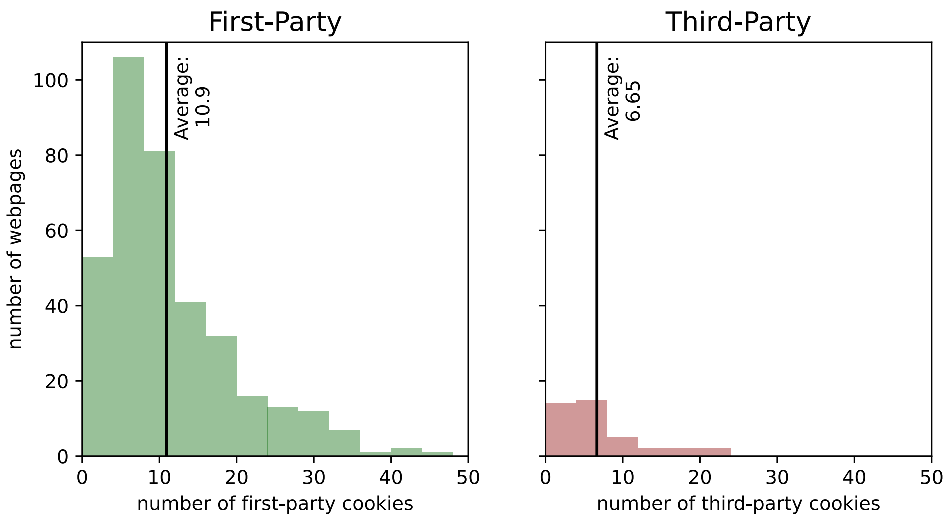


Figure 5.3: A more detailed look into first- and third-party initial cookie setting: *the distribution of the number of first and third party cookies set before any cookie dialog interactions.*

dience are more mindful about GDPR regulations, I have distinguished the average number of initial cookies that have been set by such websites. As visible from Figure 5.2, however, websites with potential European target audiences have set marginally less initial cookies, with an average of 9.42, about 2.2% less than all studied websites. Nevertheless, as seen in Table 5.1, the number of initial cookies is multimodal, with one of the modes being 0, which suggests that it may be more common for websites with European target audience to not set initial cookies based on the collected dataset.

Identifying ID-like Cookies

Sánchez-Rola et al. [36] uses a technique called `zxcvbn` to differentiate tracking cookies, which is mainly used for password strength estimation. They classify cookies as tracking which have values that have high `zxcvbn` scores. In my judgment, this measure alone, even with a high threshold, may be slightly inadequate to categorise tracking cookies. The reason for this is that a cookie having a unique and hardly guessable value does not necessarily imply that it is a user ID, and thus categorising it as tracking may be unwarranted. Another approach taken by Englehardt et al. [17], which I partially adopted in this paper, looks at various attributes of cookies. Based on their technique, I have implemented an algorithm to distinguish ID-like cookies. Firstly, the algorithm parses the cookie values because a single cookie's value often contains a list of "sub-values". These are commonly of the following form:

```
[name_1]=value_1|...|[name_I]=value_I,
```

where | stands for any character except all base ASCII Latin letters (a-z, A-Z), all digits (0-9), the underscore (_), the equals sign (=), and the hyphen (-). After parsing the cookie values, the algorithm categorises a cookie as ID-like, if it has a "sub-value" $value_i$, that has a length $8 \leq len(value_i) \leq 100$ and that expires later than 90 days from the time it is set. Such ID-like cookies are expected to be tracking cookies with a higher probability. CDA has recorded that 20.3% of websites have set an ID-like cookie on their first visit (Figure 5.2).

Looking at the Newly Collected Dataset

Table 5.1 shows that using the new somewhat inchoate method described in Section 3.3 to collect *all* cookies, has resulted in Dataset B, which has a 87.5% higher average number of initial cookies set per website. This is presumably due to the fact that a smaller portion of the scraped of websites have set a very high number of third-party cookies, as the maximum number of initial cookies installed has experienced more than a fourfold increase compared to the primary dataset. This is further deducible from the fact that there was only a small increase in the mean and the mode of the number of initial cookies. These statistics imply, that the distribution has a positive skew, similarly as for the primary dataset (Dataset A) (see Figure 5.2).

5.2 Cookie Dialogs

This section analyses the 217 collected cookie dialogs' attributes and their behaviour in relation to interactions with them. CDA has found that 49.6% of the successfully analysed websites contained cookie dialogs. These dialogs' attributes are examined in terms of the text they contain, and the main options they give users and their position on the webpage. Furthermore, their functioning is studied in terms of the number of cookies set or deleted in response to the interactions with them.

5.2.1 Cookie Dialog Attributes

CDA has collected data on the number of clickables/options cookie dialogs provide users. The data analysis has shown that majority (77.4%) of websites offer less than three options, some of which may occur to be hidden or disabled (see Figure 5.4). Additional analysis has found, however, that no correlation is present between the number of buttons and the number of initial cookies set by websites. Moreover, it was found that of the analysed dialogs 8.1% are covering a great portion of the webpage, some of which may be blocking users from interacting with the main content of the webpage (see Figure A.4 in Appendix A).

CDA has recorded all the text contained in the collected cookie dialogs. The term *dialog text* will be used to refer to these texts. Table 5.2 shows the relevant summary statistics of the dialog texts' wordcount. It can be observed that the mean value (493.47) of the wordcounts is substantially higher than the median (57). This is due to

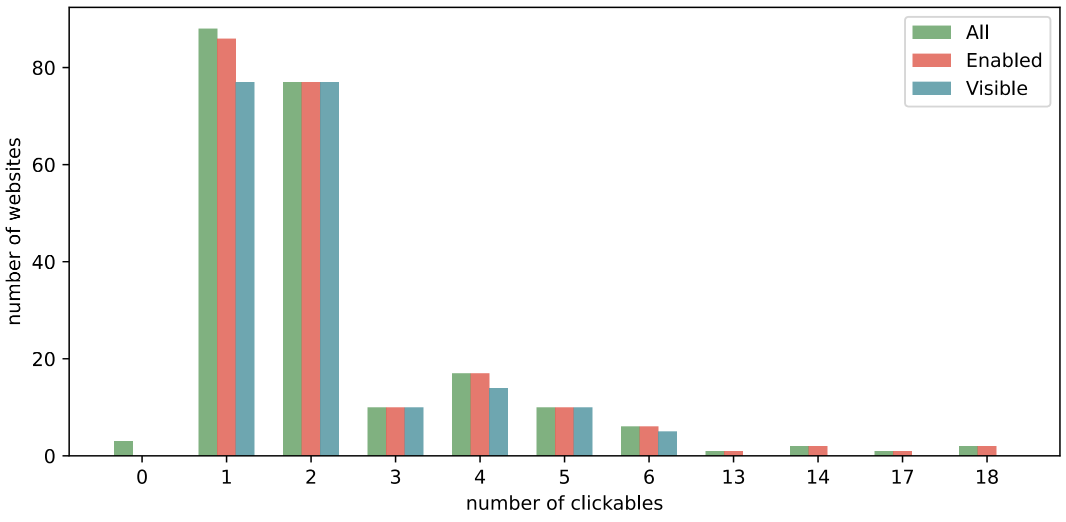


Figure 5.4: The number of websites implementing a cookie dialog with n (enabled/visible) clickables.

	Mean	Median	Max	Standard Deviation	First Quartile	Third Quartile
Word Count	493.47	57	36346	2623.08	23	110

Table 5.2: Summary statistics of the number of words in the cookie dialogs'.

the fact that the distribution is skewed by a small number of long dialog texts[2], thus, the median (57) together with the first and third quartiles (23 and 110 respectively) best describe the dialog text lengths. It can be argued that reading around 57 words before accessing the main content of a webpage may be excessive during day-to-day browsing for an average user.

Cookie Dialog Text Analysis

An important aspect of cookie dialogs is the wording used by them to inform clients about the use tracking technologies. The wording used by the notices is essential as the GDPR requires websites to use "concise, transparent, intelligible and easily accessible [wording]... using clear and plain language" [13]. My aim was to analyse what unigrams and bigrams are the most important for the dialog texts, thus, gaining insight into common wordings of dialogs.

Rank	Term	Weight	Rank	Term	Weight
1	cookie	0.369	**7**	information	0.125
2	use	0.300	**8**	policy	0.125
3	privacy	0.198	**9**	ad	0.123
4	site	0.192	**10**	content	0.118
5	use cookie	0.173	**11**	accept	0.113
6	agree	0.132	**12**	experience	0.098

Table 5.3: The *tf-idf* ranking derived from the full dialog texts.

I have used the statistical measure called *tf-idf* (Term Frequency Inverse Document Frequency) to determine which are the most common and relevant words used by dialogs [33]. Before completing the measurement, however, I had to clean the cluttered dialog texts extracted from the dialog HTMLs. The cleaning process required the tokenisation of words, meaning that all words had to be separated from all punctuations and other non-alphanumeric characters and converted to lower case. In some cases the extracted text contained words from different parts of a dialog merged together (e.g. "...privacy policyAccept..."). I have found that these merged words were often camelCase, thus I have separated such instances as well. Following the cleaning process, stop words (e.g. "and", "so") were removed and the dialog text was lemmatised, meaning that its words had to be converted to a general base form (e.g. vendors'$\Rightarrow$ vendor). Lemmatisation allows *tf-idf* to group words with the same base form together. This stage used Python's Natural Language Toolkit (nltk) for both the lemmatisation and the stop word removal [5]. Finally the dialog texts were analysed with *tf-idf*, using the *scikit-learn* library [30], resulting in a ranking of single words and bigrams a subset of which is shown in Table 5.3. The ranking shows that the most common and important words consist of mainly plain language words, which is adherent to the GDPR. Nevertheless, as shown by Kulyk et al. [26], the highest ranked word "cookie" is not well understood in this context by the web users.

[2]11 dialogs contain more than 500 words, which are likely to be privacy policies, extensive vendor lists or the combination of the two.

Besides analysing the full dialog texts, I have further done a similar investigation on the clickables' text contents. To attain a ranking of the clickables, I have followed a similar methodology as just described. The two adjustments made included not removing the stop words from the texts and converting all $\times$ symbols to the word "close". The first modification allowed the *tf-idf* algorithm to include bigrams such as "(do) *not* accept" to the rankings, which are shown in Table 5.4 (left). It can be observed that terms used to refer to opting in (e.g. accept, agree), closing the dialog, and referring to privacy policies are within the top-10 ranks.

Furthermore, the *tf-idf* ranking was also performed on clickables which are not `<a>` tags, to filter out links, which usually point to settings and privacy policies. This resulted in the ranking presented in Table 5.4 (right), which better characterizes clickables', like button tags, texts which are often used for actions such as opting in to and opting out from being tracked and closing cookie dialogs. In this ranking, clickables for opting in remain on top of the rank, while the first term, "reject" related to opting out appears at the 37th position. This indicates that refusing to accept tracking technologies may be rarely provided as an easy option for users.

Rank	Term	Weight
1	accept	0.091
2	close/$\times$	0.090
3	learn	0.084
4	cookie	0.069
5	review	0.057
6	policy	0.055
7	ok	0.055
8	agree	0.051
9	cookie policy	0.037
10	remind later	0.035

Rank	Term	Weight
1	accept	0.142
2	close/$\times$	0.108
3	review	0.098
4	agree	0.086
5	ok	0.064
6	remind	0.061
7	later	0.061
8	remind later	0.061
...	...	...
37	reject	0.004

Table 5.4: The *tf-idf* ranking derived from the all clickables' texts (*left*) and for clickables' texts whose HTML tags are not `<a>` (*right*).

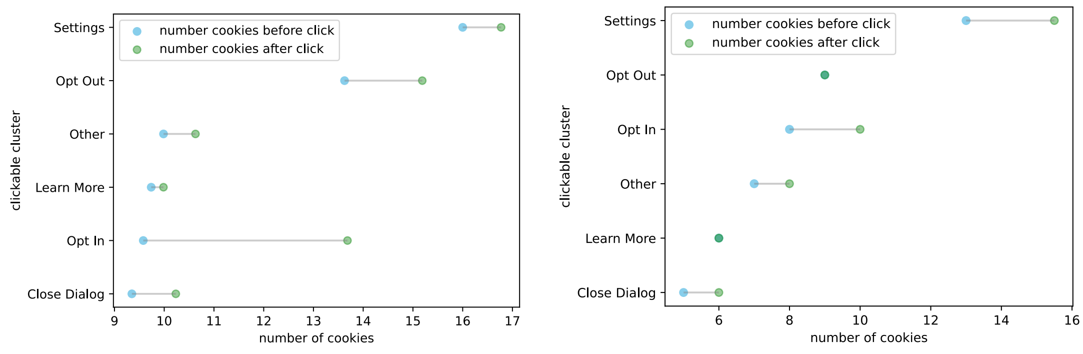
5.2.2 Cookie Dialog Behaviour

One of my further objectives was to gain an understanding about changes in the number of set cookies resulting from clicking cookie dialog buttons and links. To conduct an analysis on this subject I had to cluster the clickables based on their ostensible functions. For this, I have used the pre-processed clickable texts from the *tf-idf* analysis. Moreover, the examination required the collected initial cookies and the cookies set after individual clicks, referred to as post-click cookies.

Obtaining a clustering for such short texts is nontrivial as clustering algorithms have very little to no context to work with. As explained by Yan et al. [44], popular topic modelling algorithms such as LDA do not work well with short texts. Thus, I have experimented with Gibbs Sampling algorithm for the Dirichlet Multinomial Mixture model (GSDMM) [45], which is specifically designed for short text clustering. Yin and Wang [45] propose that GSDMM is able to infer cluster numbers automatically, which is a great advantage compared to several other well-known clustering algorithms

Opt In		Opt Out		Close Dialog	Learn More	Settings
∧		∧		close	policy	manage
¬(don't ∧ not)	agree		no	×	learn	setting
	ok		reject	exit	review	show
	get it		opt ∧ out	remind	mehr	vendor
¬(don't ∧ not)	accept		decline		read	brand
¬(don't ∧ not)	allow		refuse		term	marken
¬(don't ∧ not)	consent	don't ∨ not	accept		cookie	partner
¬(don't ∧ not)	continue		deny		change	preference
	opt ∧ in				erfahren	
	akzeptieren				privacy	
					statement	
					more	

Table 5.5: Words often appearing in clickables, grouped based on the expected function of the clickable they are contained in. The *Opt In* and *Opt Out* columns contain logical operators to define which bigrams should or should not appear together (e.g. the *Opt In* column defines that *not* and *consent* should not occur together).



(a) Mean (b) Median

Figure 5.5: Cookie setting behaviour following cookie dialog interactions: *shifts in the mean and median of the number of cookies set for websites offering clickables belonging to one of the clusters shown in Table5.5*

such as K-means. Nevertheless, the clusters of the clickable texts obtained with it have turned out to be poor and inconsistent, having numerous overlappings between them.

Hence, given the limited vocabulary of cookie dialogs, I have implemented a manual clustering algorithm which uses the grouped terms presented in Table 5.5 to cluster clickables into one of the five specified categories: *Opt In*, *Opt Out*, *Close Dialog*, *Learn More* and *Settings*. These groupings are based on the collected clickable texts. The algorithm collects the buttons and links into one of the categories by checking whether a term or a pair of words is contained in their text. All other buttons were put in the category named *Other*.

After the clustering of the clickables, I have computed the changes in the number of cookies in response to the clicks automated by CDA. Figure 5.5 shows the mean and median changes in the number of cookies set. For both measures it is visible that the click on a button or link in the *Opt In* category resulted in a substantial increase in the number of cookies installed. Furthermore, also clicking on any other option on cookie dialogs triggered new cookie placements. Figure 5.6 shows the distribution of the changes in the form of a violin plot. The diagram points out that, while the distribution of cookie changes after clicking on *Opt In* choices is centred around the highest median value of about 1.9, other choices have distributions centred around 0, with several outliers. The insights from the two figures show that the cookie dialogs induce small amounts of changes in the number of cookies, usually in positive direction.

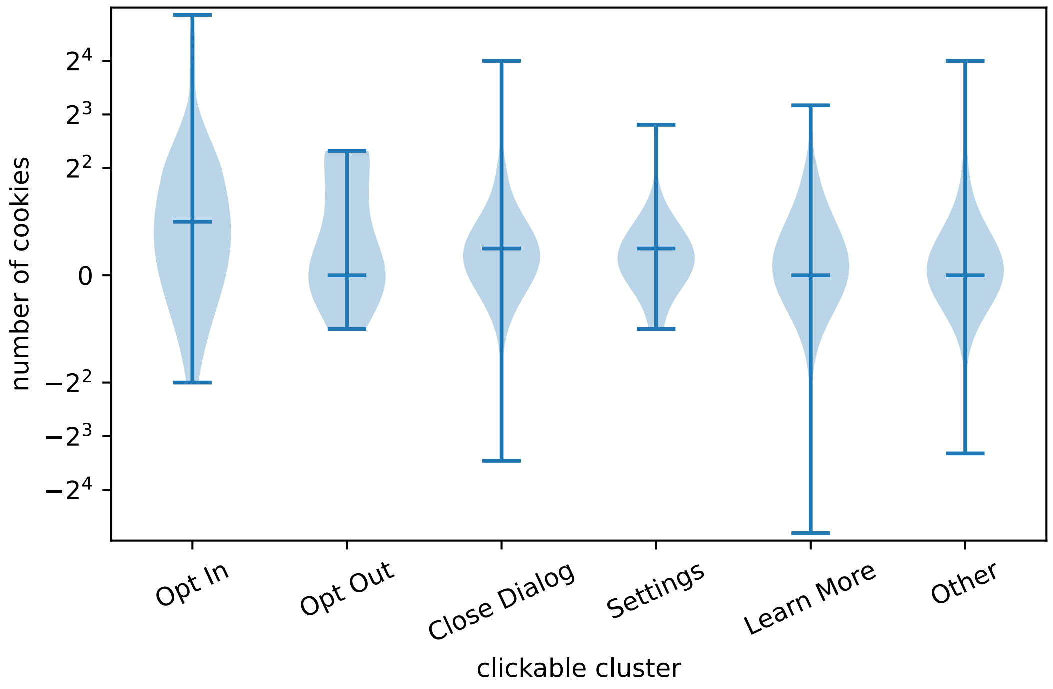


Figure 5.6: The distribution of the change in the number of cookies resulting from the interactions with the dialogs. The central horizontal lines indicate the median of the distributions. (N.B.: The y axis has a log_2 scale for values $| v |> 2^2$ to allow extreme values to fit.)

Chapter 6

Conclusion

Overall, the main goal of this study – developing and implementing a methodology for a research platform which allows the automated scraping of cookie dialogs and related web privacy data, has been successfully met. The implemented system, the Cookie Dialog Analyser (CDA), effectively overcomes the main challenges set out by the research objective. On a high level, these challenges consisted of the automatized detection of cookie dialogs on a wide variety of websites and the interaction with them. The platform has been able to analyse 444 web pages, of which 49.6% contained cookie dialogs. CDA uses Selenium, a web browser automation tool, to automate the process which was able to locate cookie dialogs with the assistance of a community-maintained filter list. Furthermore, a manual verification and correction tool has been provided to optionally fix any errors of the dialog detector. This method yielded a precision of 92.1% and a recall of 71.4% at detecting cookie dialogs. As visible from these results, the mentioned filter list has the benefit of finding cookie notices with high precision while also ensuring that the system stays up to date over time given the regular updates to the list.

The system has permitted the collection of a dataset which has been able to give insights into a segment of cookie setting behaviour before and in result of cookie notice interactions. Based on the collected cookie assemblage it has been deduced that around 20% of websites set ID-like cookies on their first visit. Furthermore, it has shown that in general all dialog interactions only marginally shift the number of cookies usually in the positive direction. The dataset further gave an overview on the options offered to users by the studied cookie dialogs. It was found that these dialogs tend to impel users to accept tracking technologies by limiting their choices, only rarely allowing them to opt out with a simple click of a button.

The built research platform intrinsically enables an easier way to conduct empirical measurements on how much control users have over their online privacy. In future work the system may be extended to capture a more comprehensive image of tracking technologies usage in response to privacy dialog interactions. The development to include all third-party cookies has been initiated, which may be taken further, while the detection of new tracking technologies, such as fingerprinting, may be additionally included. Additionally, more sophisticated methods for tracking cookie detection may also be implemented. Ultimately, such future studies may allow organisations to better enforce regulations such as the GDPR.

In conclusion, the implemented platform, the dataset collected with it and the insights from the data analysis have contributed to the area of web privacy research, as only a limited number of studies have been conducted on cookie dialogs and their behaviour. Previous research studying cookie dialogs either collected data through user studies, manually, or through an automated process targeting only well-known cookie consent libraries [28, 15, 42, 26]. The current work has contributed with a novel research platform, which automates the gathering of data, encompassing a much broader set of cookie dialogs, thus giving insight on the state of consent management of tracking technologies at a more general level.

Bibliography

[1] antvallap. *Get Unique CSS Selector*. Sept. 2018. URL: https://chrome.google.com/webstore/detail/get-unique-css-selector%5C%5C/lkfaghhbdebclkklgjhhonadomejckai.

[2] IP-API. *IP-API.com - Geolocation API*. URL: https://ip-api.com/.

[3] Artur Barseghyan. *barseghyanartur/tld: Extracts the top level domain (TLD) from the URL given*. Feb. 2020. URL: https://github.com/barseghyanartur/tld.

[4] A. Barth. *HTTP State Management Mechanism*. RFC 6265. RFC Editor, Apr. 2011. URL: https://tools.ietf.org/html/rfc6265.

[5] Steven Bird, Edward Loper, and Ewan Klein. Apr. 2020. URL: https://github.com/nltk/nltk.

[6] Ian J Block. “Hidden Whois and Infringing Domain Names: Making the Case for Registrar Liability”. In: *U. Chi. Legal F.* (2008), p. 431.

[7] Tomasz Bujlow et al. “A survey on web tracking: Mechanisms, implications, and defenses”. In: *Proceedings of the IEEE* 105.8 (2017), pp. 1476–1510.

[8] Matt Burgess. *The tyranny of GDPR popups and the websites failing to adapt*. Aug. 2018. URL: https://www.wired.co.uk/article/gdpr-cookies-eprivacy-regulation-popups.

[9] Aaron Cahn et al. “An empirical study of web cookies”. In: *Proceedings of the 25th International Conference on World Wide Web*. 2016, pp. 891–901.

[10] Richard Clayton and Tony Mansfield. “A study of Whois privacy and proxy service abuse”. In: *13th Workshop on the Economics of Information Security*. 2014.

[11] MDN contributors. *Descendant combinator*. Oct. 2019. URL: https://developer.mozilla.org/en-US/docs/Web/CSS/Descendant_combinator.

[12] MDN contributors. *Implementing image sprites in CSS*. May 2019. URL: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Images/Implementing_image_sprites_in_CSS.

[13] Council of European Union. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. https://eur-lex.europa.eu/eli/reg/2016/679/oj. Apr. 2016.

[14] Adrian Dabrowski et al. “Measuring Cookies and Web Privacy in a Post-GDPR World”. In: *International Conference on Passive and Active Network Measurement*. Springer. 2019, pp. 258–270.

[15] Martin Degeling et al. "We value your privacy... now take some cookies: Measuring the GDPR's impact on web privacy". In: *arXiv preprint arXiv:1808.05096* (2018).

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

[17] Steven Englehardt et al. "Cookies that give you away: The surveillance implications of web tracking". In: *Proceedings of the 24th International Conference on World Wide Web*. 2015, pp. 289–299.

[18] fanboy et al. *EasyList*. URL: `https://easylist.to/`.

[19] Basti G. *python selenium chrome cookie database is empty*. Aug. 2019. URL: `https://stackoverflow.com/questions/57494122/python-selenium-chrome-cookie-database-is-empty`.

[20] Michelle Goddard. "The EU General Data Protection Regulation (GDPR): European regulation that has a global impact". In: *International Journal of Market Research* 59.6 (2017), pp. 703–705.

[21] Samuel Hoffstaetter. *Python Tesseract*. Jan. 2020. URL: `https://github.com/madmaze/pytesseract`.

[22] *HTTP cookies*. Dec. 2019. URL: `https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies`.

[23] Jason Huggins. *Selenium*. `http://selenium.dev/`. 2004.

[24] *Keyword Research, Competitor Analysis, & Website Ranking: Alexa*. May 2019. URL: `https://www.alexa.com/`.

[25] Balachander Krishnamurthy and Craig Wills. "Privacy diffusion on the web: a longitudinal perspective". In: *Proceedings of the 18th international conference on World wide web*. 2009, pp. 541–550.

[26] Oksana Kulyk et al. """This website uses cookies": Users' perceptions and reactions to the cookie disclaimer". In: *European Workshop on Usable Security (EuroUSEC)*. 2018.

[27] Victor Le Pochat et al. "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`.

[28] Célestin Matte, Nataliia Bielova, and Cristiana Santos. "Do Cookie Banners Respect my Choice? Measuring Legal Compliance of Banners from IAB Europe's Transparency and Consent Framework". In: *arXiv preprint arXiv:1911.09964* (2019).

[29] mhammond and rupole. *Python for Windows (pywin32)*. Nov. 2019. URL: `https://github.com/mhammond/pywin32`.

[30] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[31] plightbo et al. *Selenium Client Driver — Selenium 3.14 documentation*. URL: `https://www.selenium.dev/selenium/docs/api/py/index.html`.

[32] Victor Le Pochat et al. "Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation". In: *NDSS*. 2019.

[33] Juan Ramos et al. “Using tf-idf to determine word relevance in document queries”. In: *Proceedings of the first instructional conference on machine learning*. Vol. 242. Piscataway, NJ. 2003, pp. 133–142.

[34] Leonard Richardson. *BeautifulSoup*. Oct. 2019. URL: `https://www.crummy.com/software/BeautifulSoup/`.

[35] *Same-origin policy*. Nov. 2019. URL: `https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy`.

[36] Iskander Sánchez-Rola et al. “Can I Opt Out Yet?: GDPR and the Global Illusion of Cookie Control”. In: *Asia CCS ’19*. 2019.

[37] *Scriptable Headless Browser*. URL: `https://phantomjs.org/`.

[38] StatCounter. *Global market share held by the leading web browser versions as of February 2020*. Mar. 2020. URL: `https://www.statista.com/statistics/268299/most-popular-internet-browsers/`.

[39] Nitasha Tiku. *How Europe’s New Privacy Law Will Change the Web, and More*. Mar. 2018. URL: `https://www.wired.com/story/europes-new-privacy-law-will-change-the-web-and-more/`.

[40] Topaco. *Chrome 80 how to decode cookies*. Feb. 2020. URL: `https://stackoverflow.com/questions/60416350/chrome-80-how-to-decode-cookies`.

[41] Martino Trevisan et al. “4 years of EU cookie law: Results and lessons learned”. In: *Proceedings on Privacy Enhancing Technologies* 2019.2 (2019), pp. 126–145.

[42] Christine Utz et al. “(Un)informed Consent: Studying GDPR Consent Notices in the Field”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 973–990.

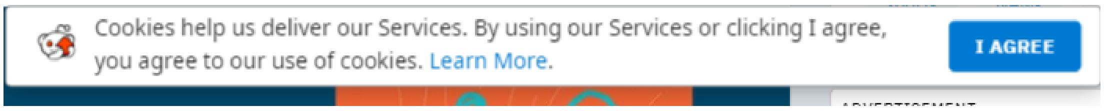
[43] *WebDriver*. June 2018. URL: `https://www.w3.org/TR/webdriver1/`.

[44] Xiaohui Yan et al. “A biterm topic model for short texts”. In: *Proceedings of the 22nd international conference on World Wide Web*. 2013, pp. 1445–1456.

[45] Jianhua Yin and Jianyong Wang. “A dirichlet multinomial mixture model-based approach for short text clustering”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 233–242.

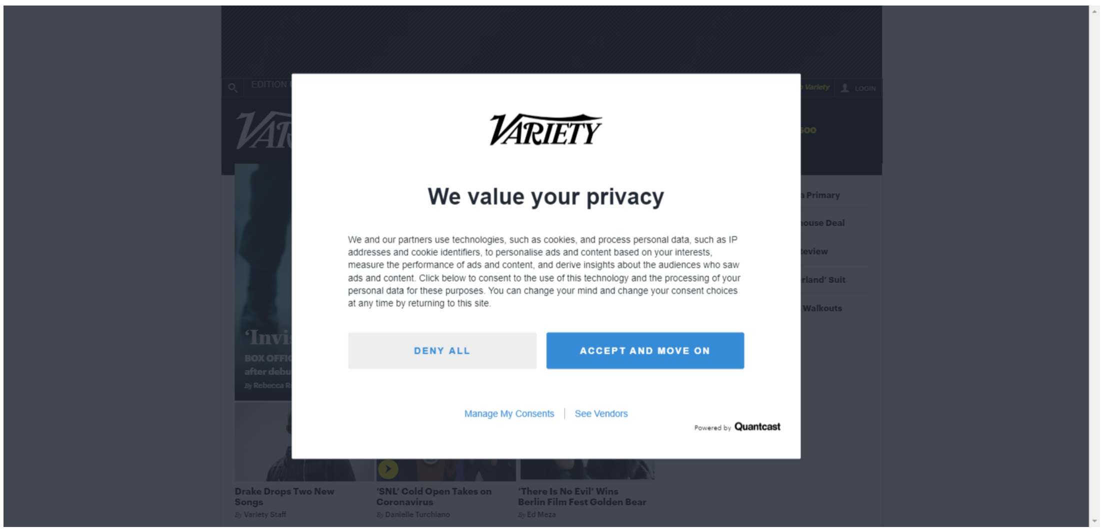
Appendix A

Example Cookie Dialogs



Cookies help us deliver our Services. By using our Services or clicking I agree, you agree to our use of cookies. Learn More.

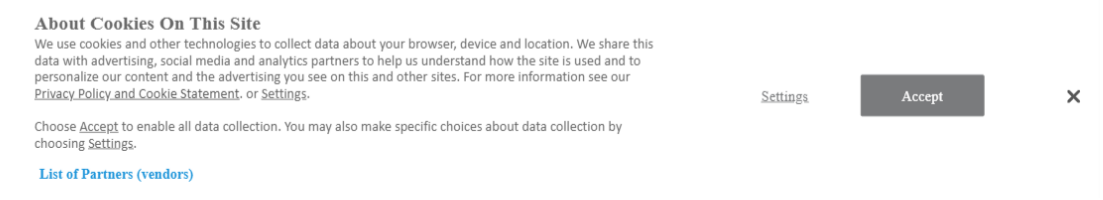
Figure A.1: An example of a cookie dialog from `reddit.com` which implies that tracking technologies are used by default.



We value your privacy

We and our partners use technologies, such as cookies, and process personal data, such as IP addresses and cookie identifiers, to personalise ads and content based on your interests, measure the performance of ads and content, and derive insights about the audiences who saw ads and content. Click below to consent to the use of this technology and the processing of your personal data for these purposes. You can change your mind and change your consent choices at any time by returning to this site.

Figure A.2: An example of a cookie dialog from `variety.com` allowing users to opt out from having tracking technologies installed on their computer. The description of what kind of personal data is collected and for what purposes might be too complex and technical for the understanding of a general audience. The dialog encourages users to opt in by highlighting the corresponding button for this action.



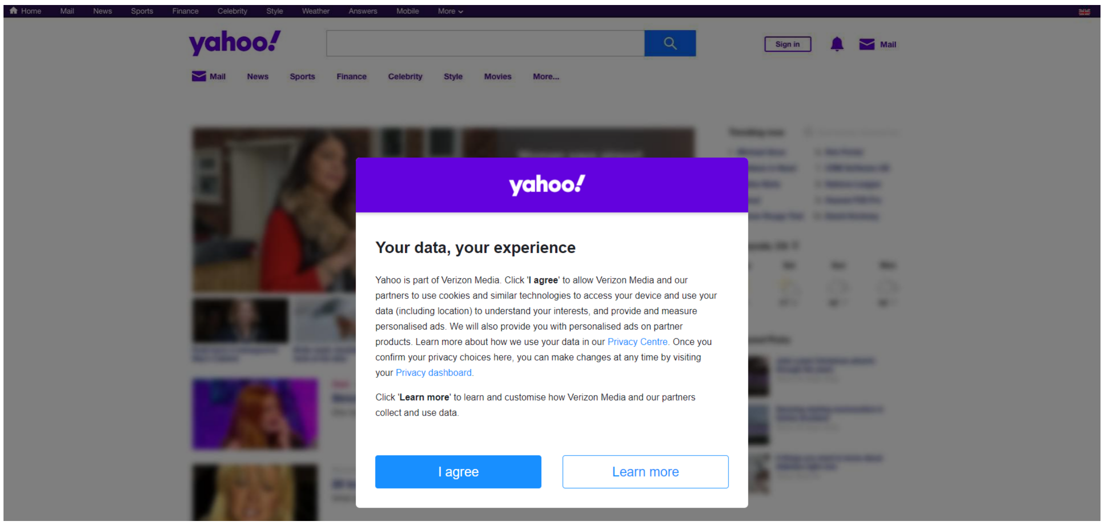
About Cookies On This Site

We use cookies and other technologies to collect data about your browser, device and location. We share this data with advertising, social media and analytics partners to help us understand how the site is used and to personalize our content and the advertising you see on this and other sites. For more information see our Privacy Policy and Cookie Statement. or Settings.

Choose Accept to enable all data collection. You may also make specific choices about data collection by choosing Settings.

List of Partners (vendors)

Figure A.3: An example of a cookie dialog from `wired.com` offering a range of options, but not allowing the user to unambiguously opt out from having tracking technologies installed.



Your data, your experience

Yahoo is part of Verizon Media. Click '**I agree**' to allow Verizon Media and our partners to use cookies and similar technologies to access your device and use your data (including location) to understand your interests, and provide and measure personalised ads. We will also provide you with personalised ads on partner products. Learn more about how we use your data in our Privacy Centre. Once you confirm your privacy choices here, you can make changes at any time by visiting your Privacy dashboard.

Click '**Learn more**' to learn and customise how Verizon Media and our partners collect and use data.

Figure A.4: An example of a cookie dialog from `yahoo.com`, which is blocking users from using the page before allowing `yahoo.com` and its partners to use tracking technologies.