Measuring the cookies behaviours under different browser configuration

Xu Zhang

Master of Science
Design Informatics
School of Informatics
University of Edinburgh
2021

Abstract

Cookies played a vital role in today's web, it offers convenience yet easily exploited by advertising companies. Even though many people do not fully understand how cookies and online-tracking work, they installed some kind of plugin, naively believing that it can protect their privacy. In this research, I have implemented a command-line tool for collecting cookie-related data and used it to investigate the effectiveness of many commonly used adblockers and other potentially cookie-blocking solutions. The results were no surprise that some of the most widely used adblockers do not offer the protection that most people believe.

Acknowledgements

First of all, I would like to thank my supervisor Kami Vaniea for all her support and encouragement throughout this project.

I would also like to thank the everyone in the Tulip weekly meeting.

Finally, I would like to thank all my friends for keeping me sane throughout COVID and my project.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Xu Zhang*)

Table of Contents

Chapter 1

Introduction

1.1 Motivation

In the last two decades, the Internet has grown exponentially, with its user increased from 413 million in 2000 to more than 4.5 billion in 2020[1].

Every user browsing on the internet are leaving digital traces on the websites they visited. Upon each visit, mouse and keyboard action, user's personal data e.g. preferences, habits are being collected and shared between advertising companies. These data collected are highly valuable however this act is a breach of the privacy of the users. There are many tracking technologies used e.g. Third-party cookie tracking, cookie synchronization and browser fingerprinting. The most commonly used method is Third-party cookie tracking.

In the past, due of the lack of regulations, websites set tracking cookies without consent from the user, a study done by PwC and UK government [2] in 2011 suggested that 87% do not fully how cookie works and 37% of user do not know how to manage their cookie preferences. To combat privacy issue with this practice, the General Data Protection Regulation (GDPR) was introduced in 2016. This regulation requires websites with users accessing from European Union and European Economic Area to prompt some form of cookies warning and allowing user to control their cookie preferences.

Although research [2] suggest most people do not understand cookies, it also found that 77% of the users are concerned about internet security and privacy. This is potentially one of the reasons why ad-blockers are very popular, a previous survey [3] suggested that 42% of user have some form of ad-blocking tool installed. These adblockers claimed that they protect the users from being tracked by filtering out third-party

HTTP request, However, a study [] done in 2016 suggested that at default setting, adblockers are not actually doing much with some adblockers like Ghostery have almost no effect at default setting.

Another viable cookie-blocking option that many users not aware of is AdChoices or OnlineChoices provided by DAA[4] (Digital Advertising Alliance) and EDAA[5] (European Interactive Digital Advertising Alliance). AdChoices is a method allowing user to opt-out cookies from hundreds of participating advertising agency in one button click. There is limited research done on the effectiveness of this method.

All of these raises some interesting research questions. For example, it is now 3 years after GDPR in enforced, are there more websites comply with the cookie law, did adblockers improved in any way to match user expectation 4 years after previous effectiveness study, is AdChoices a good alternative to adblockers, or it is just a "solution" for advertisers to maintain self-regulated?

This research aims implement a privacy measurement tool that can automatically crawl through a set of domains and collect cookies related data while under different browser configuration. I will be using the tool to collect data and investigate the effectiveness of various cookie-blocking options and the behaviour of the cookies under each option.

1.2 Hypothesis

- Mass cookies opt-out should eliminate all third-party cookies, at least third-party cookies from DAA participating advertisers
- All adblockers investigated at default setting are effective in cookie-blocking to certain extent due to the "acceptable ads program" , and there is a significant increase in effective as protection level is increased
- As the protection level increases, more webpage content will be affected and may lead to losing some functionality of the website

1.3 Contribution

I conducted investigations on the Top 1000 domains from the Tranco List[6] and the result suggest that Adblock and Adblock plus at default, medium provide unsatisfiable performance and only provide significant protection max out all protection options.

Ghostery which did poorly in the previous study did surprisingly well this time, even at default it offers comparable performance to Adblock(Max out).

The mass opt-outing using DAA/EDAA method have about the same affect as default/medium setting of Adblock and Adblock plus. This performance is unsatisfiable considering the effort required to use them.

- Developed an automated crawling tool based on Selenium that simplified the cookies and cookies-related data collection process
- Collected cookies data from 1000 domains under different browser configuration
- Visualized the collected data to gain some insight of the cookie behaviours and draw comparison between different cookie-blocking tools

1.4 Structure

This thesis is consisted of 7 chapters.

- Chapter 2 will be introducing all important background knowledge required to understand various concept used and the experiment outcome
- Chapter 3 explain the design of the measurement system, how it is implemented and justification behind each design decision
- Chapter 4 will be showing all the the details of the experiment setup used for this study.
- Chapter 5 outline the results of the experiments and visualized the data to help draw comparison,
- Chapter 6 will be discussing potential future works, many of which is not done in this research due to time limitation
- Chapter 7 will be concluding the outcome of this research and the recommendations to users and policy makers that derived from experiment outcome.

Chapter 2

Background

2.1 Web cookies

HTTP cookies or Web cookies [7] were first introduced in 1994. This mechanism was initially implemented to enable state-maintaining between clients and the server. When a browser makes a GET request to a server for a webpage, the server responds with the webpage content and cookies in the header that uniquely identifies the user and their machine. Cookies are formatted strings consisted of key-value pairs separated by semi-colons, Figure 2.1.

Figure 2.1: Basic Cookie setting and using process

A example of cookie can look like this **(Name=Value; Host=examplesite.com; Path=/home; Expires=1 May 2021 11:13:05 UTC; Secure)**. A cookie can have following attributes[8][7]:

- **Name:** A name that can be any US-ASCII characters, except some special character and separator characters. It is given by the server that sent the cookie and

can be used to uniquely identify the cookies for the server.

- **Value:** The value that pairs with the cookie name, it is a randomly generated string of US-ASCII characters excluding some special characters, this value is the main data that cookie carries, it can be used to remember site setting, preference and mostly importantly it can contain Identification string that uniquely identify the client, this value can be clear text but is usually encrypted for privacy and security.

- **Domain (Host)** This attribute specifies the origin of the cookie, the website that set this particular cookie and browser uses it to determine and send correct cookies when performing subsequent requesting for any websites.

- **Path:** This attribute determines at which sub-directories of the website a browser should sends the cookie in the header. For example, a cookie set by **(exmaple.com)** with cookie path **(path=/catpath)**, then when visiting **example.com/catpath** or **example.com/catpath/111** or any sub-directories matching path pattern, browser will send the cookie in the header

- **Expires:** This attribute value represent the lifetime of a cookie, this value is usually in epoch time also known as unix timestamp which looks like a integer string, this time is relative to the client, not the server setting it.

- **Max-Age:** This attribute is similar to Expires, it represent number of seconds until the cookie expires. In some cases where Expires and Max-Age are both present, Max-Age has precedence.

2.2 Cookie types

Web cookies generally fall under two categories, First-Party and Third-Party cookies,

2.2.1 First-party cookies

First-Party Cookies [9][10] are cookies created and set directly by the domain(Host) that the user is visiting. For example, if user A made GET request to **"www.google.com"**, then cookies with the domain attribute **".google.com"** are the First Party cookies. Although in some cases, when a company owns multiple domains, it starts to get ambiguous. For example, when visiting **"Microsoft.com"** some cookies belongs to

".live.com" and vice-versa. Regardless, these cookies are generally used for session management and personalization, e.g. Login status, language setting, shopping cart.

There are two different types of First-Party cookies, session and persistent, each for different tasks.

2.2.1.1 Session

If a First-Party cookie does not have a **Expires** attribute, it is considered as a session cookie. Session cookies are only stored in the memory(RAM) of the system and get deleted permanently once the "current session" is ended[11]. As browsers define when the "current session" ends, cookie deletion timing varies but usually happens when the browser is closed. However, browsers like Chrome have "restoring session", which may cause session cookies to last indefinitely.

Session cookies allow the website to remember the user from page to page, hence it is primarily used for session management tasks like maintaining shopping carts for a visitor that are not meant to last longer than one browsing session.

2.2.1.2 Persistent

On the other hand, if a First-Party cookie does have a **Expires** attribute, then it is a persistent cookie. Unlike session cookies, persistent cookies are stored locally in the user's device, and they are only deleted when the specified expiration date or Max-Age is meet.

Persistent cookies are used for maintaining login status, website analytic and website preferences , e.g. language settings. Unlike shopping cart sessions, these cookies are usually maintained for a much longer time for better browsing experiences and counting unique visitors[11].

2.2.2 Third-Party cookies

Third-Party Cookies [9][10] are cookies created and set by the domain(Host) different than the one user is visiting. For example, suppose user A made GET request to **"www.google.com"**, and a Javascript within the current website made another GET request to a third-party service "DoubleClick". In that case, the user will receive cookies set in the header of responses from both GET requests. Cookies that have domain **".doubleclick.com"** are considered as Third-Party cookies for the current website.

Third-Party cookies are usually for user tracking by including an ID-like string in its value attribute. This ID-like string is unique and can be used to identify a user from site to site to track and record user behaviour and interest. Using this information, advertising agencies can push more personalized advertisements to achieve interest-based advertising. Third-Party cookies can also be used for session maintaining of the third-party service, in this case, there should not be any ID-like string present in the cookie for user tracking

2.2.3 Identification cookies (ID-like cookies)

Identification cookies[12] are those cookies that have some form of ID-like strings present in the value attribute and its usually long-lived for longer user-tracking. First-Party cookies can also be identification cookies, but as these cookies are not shared outside the domain user is visiting and are usually used for the sole purpose of login and authentification, there should not be any privacy concerns. However, third-party cookies that have ID-like strings that are almost solely used for cross-site user tracking and advertisements[10].

2.3 Cookie blocking options

There are many options for a user to block cookie, some of the popular choices include various Ad-blockers and cookie opt-out.

2.3.1 Ad-blockers

Adblockers is the term given to a collection of software that has the ability to block advertisements shown on a website. As cookies are created in the response header after a GET request, many of the Adblockers works by using community-maintained filter rules. These filter rules contains domains and CSS selectors which Adblocker check against use to block HTTP request and block certain HTML element from showing.

2.3.1.1 Do-Not track

Two of Adblockers investigated in this study works fully only when ”Do-Not track” option of the browser is enabled.

Do-Not track(DNT) is a field in the HTTP header proposed in 2009 to allow users to indicate their preference of opt-out tracking, effort was made in Wide Web Consortium(W3C) in 2011 to standardise DNT between browser and servers, and now most modern browser supports DNT natively. However, due to the lack of support, DNT is not widely adopted by the advertising industry, with W3C closing DNT working group and Apple discontinuing support for DNT in Safari Browser. Adblockers claims that having this feature enabled in the browser, they can allow ”ads without third-party tracking” to display and create a middle ground between user’s privacy and website revenue.

2.3.2 Digital Advertising Alliance

The Digital Advertising Alliance (DAA) [4] is a non-profit organisation that was founded by major advertising and marketing trade associations.It was founded in October 2010 with the goal of developing self-regulatory standards for the online advertising sector.

DAA is global alliance but with primary focus in the USA and Canada, as privacy law in the US is significantly different from Europe. Europe launched their own version of DAA, the European Interactive Digital Advertising Alliance(EDAA)[5], this alliance develop the self-regulatory program specifically for Europe to comply with GDPR

Both DAA and EDAA provided a tool(YourAdChoice[1] and YourOnlineChoices[2]) allowing user to modify and view the cookie opt-out status of all participating advertising agencies from a single source.

2.4 The General Data Protection Regulation

The General Data Protection Regulation (GDPR[3]) is a regulation in EU law on data protection and privacy for European Union and European Economic Area. The GDPR was first introduced in April 2016 and became enforced in May 2018. The cookie compliance section of the GDPR requires all website to prompt a cookie dialog that allows user to control their cookie preferences, must not place any cookies before receiving a consent from user and providing a functioning website even if user refuse use of certain cookies[9].

[1] https://youradchoices.com/
[2] https://www.youronlinechoices.com/uk/
[3] https://gdpr.eu/

2.5 Related work

There many researches done previously on web cookies, adblockers and web privacy in general, many of the previous studies conducted on the top website from Alex ranking

2.5.1 Online tracking

Englehardt and Narayana[13] conducted a detailed measurement of online tracking pre-GDPR. They made a crawling tool, OpenWPM based on Selenium and used it to crawled the top 1 million websites and made 15 different measurements. They found that in the 90 million website requests made, the total number of third-party trackers on a minimum of two websites were over 81,000, but with the majority of these trackers present on less than 1% of all websites, suggesting that most third-party trackers an average user can encounter on a daily basis belongs to a small group of the organization.

This finding is justified by the research done by Krishnamurthy et al[14], where the number of distinct third-party trackers decreased, and a very small number of companies tracks the user across almost all popular websites.

Another research by Hu et al [15] on measuring the interconnectedness of the third party ecosystem also suggest that the third-party ecosystem is highly interconnceted by a few of large companies

2.5.2 web cookie

An empirical study of web cookies [7] crawled top 100K website from Alexa ranking and collected 3.2 million cookies. It was found that Third-Party cookies dominate First-Party cookies in terms of numbers, but a majority of the cookies were set for performance assessment(site usage statistics) purposes where it does not expose any personal information. However, nearly 80% of cookies were sent with maximal permission insecurely, hence vulnerable for cross-site attacks.

Trevisan et al. [16] conducted a cookie study on 35,000 websites from 2015-2018 to check the implementation status of the EU cookie directive. They found that about 49% of the website does not comply with the EU cookie directive, even after GDPR was enforced in May 2018, no significant change was observed. They have also found that about 28% of the websites do not show any form of cookie dialog while still installing tracking cookies, and the ones that had a cookie dialog, 93% installs some

form of tracking cookie before any user consent is made.

Dabrowski et al. [17] conducted another research on web cookies Post-GDPR in 2019, they collected cookies from Alexa Top 100,000 websites, they compared cookie setting between EU and US visitors as well as comparing cookie setting behaviour between 2016(GDPR first introduced) and 2018(GDPR being enforced). It was found that EU visitors encounter significantly less persistent cookies compared to US visitors 49.3% of top 1000 website and 26% of the top 100,100 websites refrain from setting cookies to EU visitors without consent. It also found that from 2016 to 2018, around 10.3% and 24.3% of the Top 1000 website and 30.88% and 46.7% of the Top 100,000 stop setting persistent cookies both are showing the effect of GDPR. This study however only focued on First-Party cookies and it colletected cookies using a headless browser which can affect number of cookie collected shown by Englehardt et al. [13]

2.5.3 Adblockers

Two surveys done by Arunesh Mathur et al.[18] discovered that most users and non-user of adblockers only have a basic understanding of online tracking. Their survey also shows that most adblockers users remain in default mode after installation, with only 10% Adblocker users enables higher protection (e.g. EasyPrivacy).

Arthur Gervais et al. [19] developed a quantitative framework for comparing the privacy provision of adblockers in 2016. They have then used the framework to compare AdBlock Plus and Ghostery. They found that Ghostery at default setting provides the same level of protection as with no adblocker installed. However, at maximal protection level, Ghostery is the better performing adblocker compared with AdBlock Plus.

Chapter 3

Measurement tool

The primary goal of this study is to examine cookie behaviour in various browser setups. To achieve this goal, I would have to manually collect cookies from thousands of websites, which would take a long time and be prone to errors. As a result, I created and deployed a Selenium-based command-line application that simulates real-time human web browsing while collecting cookies and other associated data.

This chapter will lead you through the system's requirements, design, and decision-making process during implementation.

3.1 System design

If time allows, there are several potential study routes to take the project farther. As a result, I divided the system's functionality requirements into minimal and supplementary requirements.

3.1.1 Minimal functionality requirement

The minimal functionality requirement specifies the tool's basic features that enable me to complete the base project. Its feature includes the followings

1. Accept command line arguments that specifies environmental variables
2. Read a list of domains acquired from Tranco list
3. Download CRX file of the adblockers from Google Chrome[1] Webstore and install to the Selenium webdriver

[1] https://chrome.google.com/webstore/category/extensions

4. Depending on the command line arguments, automate the process of changing settings of the adblockers or cookie opt-out with DAA/EDAA
5. Visit each domain from the list and collect cookies, installed cookies are cleared between each domain.
6. Distinguishing First-Party and Third-Party cookies

3.1.2 Additional functionality requirement

If time permits, more improvements to the system can be implemented. These enhancements would improve the system's usability, efficiency, and robustness, as well as allowing investigation of additional interesting cookie behaviour scenarios.

1. Record number of HTML element present in the current webpage
2. Work out number of JavaScript element present
3. Crawl session saving and restoring in case of total system failure
4. Cookie dialog interaction to simulate more realistic browsing process
5. Replace command line tool with GUI

3.1.3 System Overview

The system overview flowchart is shown in Figure 3.1

3.1.4 Top websites

A decision to be made prior the implementation is how many and what websites should be used for crawling and cookie analysis. There commonly used website popularity ranking in many other studies includes Alexa Top Sites[2] and Cisco Umbrella [3]

Alexa is Amazon Web Service (AWS) that provides lists of most popular websites according to Alexa Traffic Rank algorithm. This is usually the go-to option, However, Alexa stop updating their free CSV and is now a paid service, free tier user can only view the top 50 website, which is too little for our usage. As this research will be undertaken over around 2 months, paying $150 monthly fee is not a viable option.

[2]https://www.alexa.com/topsites
[3]http://s3-us-west-1.amazonaws.com/umbrella-static/index.html

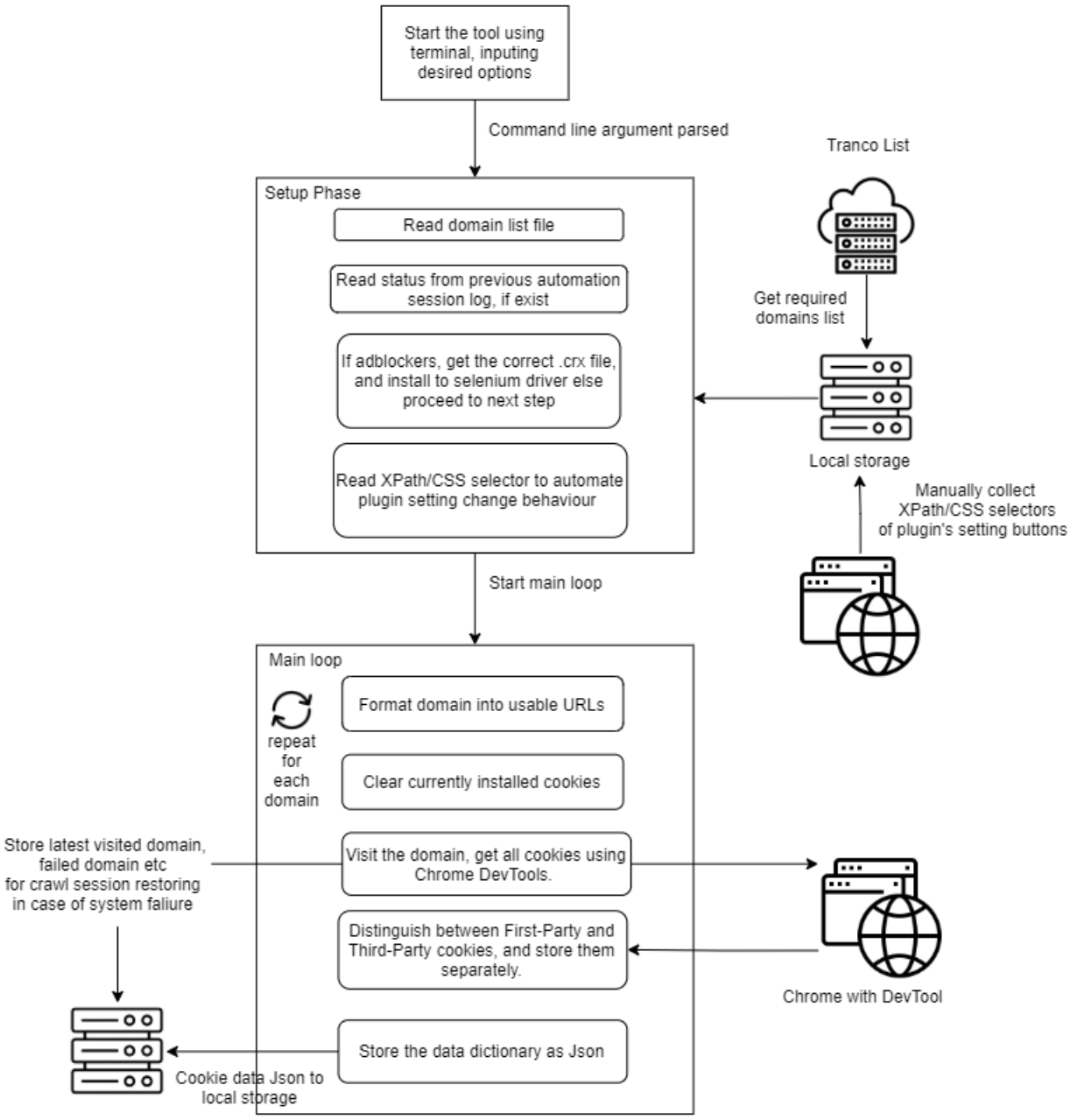

Figure 3.1: Overview of basic functionality of the system

Cisco Umbrella Popularity List on the other hand is free option that provides Top 1 million most popular websites, it calculates the rank based on the DNS traffic to their DNS resolvers (OpenDNS), this method of collecting means that non-browser traffics are also recorded and therefore may include invalid domains in list.

Furthermore, both options have been shown in a study by Pochat et al[20], that these commonly used website ranking service can be easily manipulated adversaries and therefore would greatly affect the outcome of any research study using them. So,

I will be using the Tranco list proposed by Pochat et al, it is claimed to be a improved website ranking that combined four list(Alexa, Cisco, Majestic and Quantcast) while filtering out unavailable or malicious domains, providing better stability and much more resilient against manipulation.

To ensure the research can finish within the limited time, I have chosen to use only the top 1000 websites, this should cover what most user are visiting daily.

3.2 System implementation

3.2.1 Browser automation

As mentioned previously, there already exist many web-browsing privacy research tools like FPDetective[21], which utilises PhantomJS and Chromium. Although it is specifically designed to detect and analyse browser fingerprinting, it offers customisation, so it can easily be extended for non-fingerprinting studies. However, since it only supports stateless measurement and PhantomJS is discounted for years now, the tool is very outdated and not suitable for my need.

Another tool used in several pieces of research is OpenWPM[13]. This tool utilises Selenium for web crawling and should be ideal for our usage in cookie study, as proven by other studies. However, as the tool is built for comprehensive web privacy researches, the codebase is a bit too complicated for extension, and since I will likely be interacting with the webpage DOM (e.g. cookie dialog interaction), it is probably best to build a tool specifically designed for my need.

As Selenium is widely used to automate ”real” browsing to test webpages, and many learning resources are available. I have also decided to base my implementation using Selenium.

Selenium[4] supports most of the modern browsers, e.g. Edge, Firefox and Chrome. To achieve the most ”realistic” automated browsing, I will need to matches the browsing environment of most users. According to market share research, Chrome owns 65.13% of the browser market worldwide, several times more than other browsers. Therefore to I have decided to used Google Chrome (version 91.0.4472.101) for the current implementation. Another reason for choosing Chrome is the capabilities of Chrome DevTool, with Selenium 4 supporting Chrome DevTool natively[22], this will significantly help the implementation process. Although I have chosen Chrome for this

[4]https://www.selenium.dev/

implementation, this is no means of a fixed design, as Selenium's built-in functions and methods are almost universal across different browsers; one can easily replace Chrome with any other browser.

3.2.2 Adblocker setup automation

3.2.2.1 CRX downloading

The first part of the implementation is to install different adblockers. One way to install plugins for Selenium drivers is to make Selenium explicitly wait for a couple of minutes and letting somone manually install desired plugins through the chrome web store, then proceed further to crawling. This is not automated and not ideal. Therefore I will need to install the plugins using CRX files, which Selenium can use to initialise the webdriver with plugin pre-installed. CRX files can be manually obtained using online extractors, but downloading them manually means that the plugin will get outdated in the future, so I have automated the CRX file obtainment process. The process is automated using a "backdoor" of the Chrome Webstore described here[23], this still requires someone to collect IDs for the plugins manually, but as long as it is not changed, my implementation can download the latest plugin CRX file.

3.2.2.2 Plugin setting change automation

One core objective of this research is to test the effect of different settings of the adblockers on cookie behaviours. To ensure consistency of the setting environment made, I have also automated this process, where the settings change of the adblockers can be done through command line arguments.

In Chrome, if an installed plugins support setting changes, then it will have a dedicated setting page that can be visited with the URL of the format **(chrome-extension://pluginID/options.html)**. This is just a URL like all the others that Selenium can visit using **webdriver.get()**

Initially, I tried to search for clickable buttons in the setting page manually using XPath, this was then proven very difficult and time-consuming. I then discovered the DevTool capabilities of Chrome. As shown in Figure 3.2 , I can manually tag a button and copy its XPath/CSS selector string. This selector string can be used in Selenium with the method **find_element_by_css_selector**. Once the button element is located, simply call **click()** to tick that desire option.

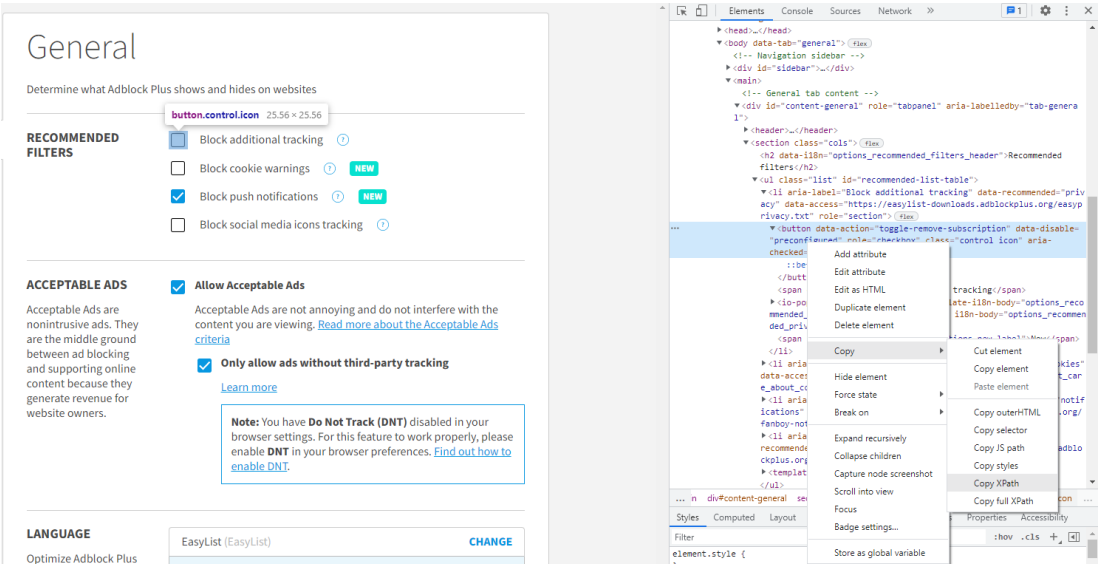

Figure 3.2: Chrome devtool usage

I collected selectors for all button/tickbox elements of the three adblockers used in this project and stored them locally, the system can retrieve the selectors when needed.

Using the above method, I have assumed that the adblocker's setting page layout does not often change, which probably wound change sometime in the future and break the system. However, I was not able to find any permanent solutions that can automate webpage DOM interactions, and the proposed method is the simplest for anyone using/updating the selectors in the future.

3.2.3 DAA opt-out automation

Cookie opt-out using DAA is a much simpler process. One can visit the AdChoice website, click few buttons, wait for a couple of minutes, and it is all set. Despite the simplicity of the process, some participating advertisers fail to respond to cookie opt-out requests and requiring the user to repeat the mass opt-out procedure.

For the reason above, I have implemented the cookie opt-out automation into two modes, fully automated and human-assisted setup. For the fully automated mode, I have adopted the same method as in section 3.2.2.2, where I manually collect CSS selectors for the interactive buttons on AdChoice. The implemented system will click the buttons in a predefined sequence with some waiting in between. The in-between waiting time of the button clicks are set to 2 minutes, this is significantly longer than what each step of the opt-out procedure(30 seconds maximum for standard network speed) takes, so in case of a slow network, the system would still function properly.

The above automation is assuming that no repeating is required, which is almost never the case with AdChoices. Therefore, I have also added a human-assisted mode, where the system loads the environment variables and navigate to AdChoices. It will then gives the user 10 minutes to apply the AdChoice opt-out manually. As AdChoice cookie opt-out usually takes no more than three attempts, 10 minutes is more than sufficient to complete the task.

Regardless of how many attempts were made, some participating advertisers fail constantly. Therefore, for both mode, before moving into the crawl phase, the system identifies those failed by using element class (the class were unique for columns of the page Figure 3.3) to locate row element and check the status text (Opt-Out Temporarily Unavailable Figure 3.4) and taking a screenshot of those rows for later analysis.

```
▼<div class="member-row" ng-class="{open: member.collapsed}"> == $0
```

Figure 3.3: Row element class

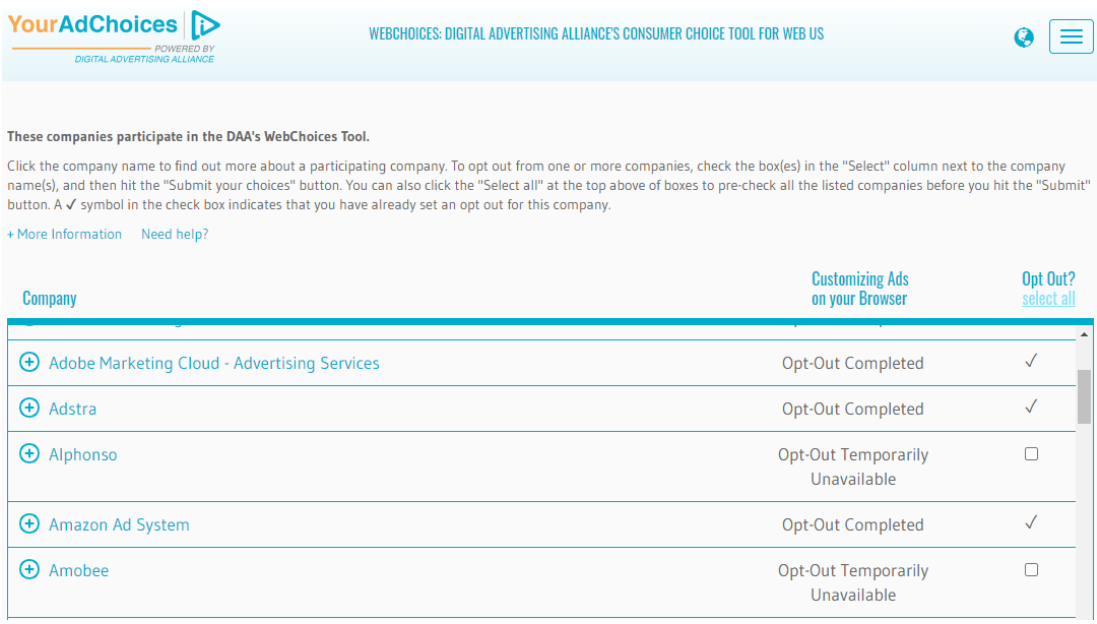

Figure 3.4: AdChoice Interface

3.2.4 Webpage crawling process

3.2.4.1 Visiting websites

Website visiting using Selenium is trivial, simply pass the desired URL into method (webdriver.get()). However, the format of the URL accepted by Selenium is restricted.

Selenium only allows URLs with the following format

1. https://www.google.com
2. https://google.com
3. http://www.google.com
4. http://google.com

Domains in the Tranco list are only avaliable in the format of **domain.com**, so I will need to reformat these domains for Selenium. In 2021, despite many browsers are enforcing HTTPS Web Protocol[24], there are still many old websites uses HTTP only. When visiting a website using a URL starting with ”http” and the website supports HTTPS, Chrome will force to redirect the URL visited to ”https”. Due to this behaviour of the Chrome browser, my implementation will be checking formats 3 and 4 only and relying on Chrome redirecting URL visited where necessary.

3.2.4.2 Webpage failure and timeouts

The hosting servers of the websites can experience downtime due to various reasons, e.g. power outage and system crash, and there is no way to know the status of the hosting beforehand. Therefore, during the crawling process, website failures are expected and should be handled.

Website failures can usually occur in the form of different HTTP status codes, DNS lookup failures, long loading(not responsive).

To exclude these erroneous websites and to prevent the crawling system from crashing, I have added a page load timeout of 120 seconds. The **webdriver.get()** method of Selenium waits until website’s onload event is fired, indicating that the webpage is fully loaded[25]. In an event of loading failure, onload event will not fire, Webdriver.get() halt the crawling process, and once the timeout threshold is met, a TimeoutException is thrown, crawling system record the failed domain and proceed to the next one.

However, because the presence of AJAX(Asynchronous JavaScript and XML), Selenium cannot know if all JavaScripts are fully loaded when onload event is fired[25]. Since some cookies are set by JavaScripts, between each page visit, after onload event is fired, system will wait for another 30 seconds to ensure all JavaScripts are loaded, I have assumed that the AJAX that do not load in 30 seconds are broken.

Furthermore, although browser automation enabled by Selenium is realistic, it is still detectable by website shown by others[26]. One website(bet365.com) out of the top 1000 we crawled manage to detect the presence of Selenium and refused to load and causes the system to halt indefinitely and ignoring the preset page load timeout. I have not yet find the cause of this behaviour, therefore it is expected that more websites can cause this behaviour. It is hard to predict all types of the website that can crash the crawling system, therefore while using the system, a user should identify the problematic websites and excluding them from the crawling list manually.

3.2.4.3 Cookie collection

Initially, I expect the cookie collection process to be straightforward using the built-in method of the Selenium **webdriver.get_cookies**.

The documentation of Selenium[27] states that the **get_cookies** method ”Returns a set of dictionaries, corresponding to cookies visible in the current session”, I was expecting to get all cookies(First and Third-Party) for the current session. However, after some experimentation, it is confirmed that get_cookies only returns First-Party cookies of the webpage, this is also confirmed by Molnar’s work [28].

The Chrome DevTool mentioned in the previous section supports a list of commands specified in their protocol[29]. There are two commands that can be used to retrieve cookies, **Network.getCookies** and **Network.getAllCookies**. As the documentation states, The **Network.getCookies** command retrieves the cookies for the ”current URL”, a similar description as the built-in method of Selenium, and I was expecting it to retrieve all cookies set after visiting the current webpage. However, it seems that this still only retrieves the First-Party cookies.

The second command Network.getAllCookies claims to retrieve ”all browser cookies”, which it did. Therefore, since I have all cookies installed and First-Party cookies for the current webpage, I can remove First-Party cookies from ALL cookies to get the Third-Party cookies given there are no cookies set by other websites, which is not a problem since we are clearing cookies between each website visit.

3.2.4.4 Cookie collection for DAA opt-out scenario

DAA cookie opt-out works by installing a set of ”opt-out” cookies to the browser, these ”opt-out” cookies are sent in subsequent requests to other websites indicating that the user is already opted-out.

This means that we cannot clear all browser cookies between each website as it will clear all the ”opt-out” cookies. Therefore, once DAA finishes installing the ”opt-out” cookies, I captured and stored these cookies locally.

Between each website visit, all cookies are still cleared, but this time, after cookies are cleared and before it moves to the next website, all ”opt-out” cookies are installed back for the browser. This way browser is in the same state as it finished opt-out using DAA initially.

3.2.4.5 Webpage element collection

After cookies are successfully collected, the system proceeds to collect two more data, the number of HTML elements and JavaScript elements.

Collecting webpage elements are straightforward using Selenium. To finding all elements regardless of the type, I used XPath (”//*”) where * simply matches everything in the DOM.

In HTML code, Javascript exist between the <script>tag, Therefore to identify javascript, I used Selenium to find all elements with tag name “script”

3.2.4.6 Data storing

The whole crawling process is time-consuming, it is prefered not to repeat it unless necessary, therefore all data collected are stored locally for long term usage.

To help with later analysis, I have adopted to store collected using HashMap, better known as Dictionary in Python. I have chosen this data structure as the access time of Dictionary is O(1)[30], this quick access time can be significant if the research is done at a much larger scale, e.g. 1 million websites. Furthermore, Python supports direct conversion from dictionary to JSON and vice versa, this file format makes the manual observation of the data collected much easier.

The URL domains visited will be used as the first-level key and following by ”first_party”, ”third_party”, ”html_elements” and ”js_elements” as the second-level keys with data are stored under their corresponding name shown in Figure 3.5

3.2.4.7 Session restoring

As this implementation is a prototype and is not test extensively. Therefore, occasionally the system can experience some problems that lead to a complete system crash.

```
{
  "google.com": {
    "first_party": [
      {
        "domain": ".google.com",
        "expires": 2145916800.934384,
        "httpOnly": false,
        "name": "CONSENT",
        "path": "/",
        "priority": "Medium",
        "sameParty": false,
        "secure": true,
        "session": false,
        "size": 18,
        "sourcePort": 443,
        "sourceScheme": "Secure",
        "value": "PENDING+290"
      }
    ],
    "third_party": [],
    "html_elements": 327,
    "js_elements": 15
  },
```


Figure 3.5: Parsed JSON view

A general usage scenario is to set up the system and letting it run for an extended period of time. If the system crashes entirely during the process, we risk losing all collect data and require repeating this time-consuming process.

Therefore, to avoid this issue, for each website crawled, data collected is immediately stored to local storage, and the website visited most recently is also constantly kept in a **recent.txt** file. If the system crashes for any reason, the next time the user restarts the system, and the system detects the presence of **recent.txt**, it will read the file and continue the crawling from this website onward.

3.2.5 Cookie Dialog interaction

This is an additional implementation on the base design. It is made to investigate if opt-in using cookies dialogue affects cookies installed when adblocker is enabled in

the background. This is a more realistic browsing scenario, where even adblocker is installed, a user can still occasionally click on cookie dialog just to get rid of its presence.

As websites update layout and style very frequently, identifying cookie dialogs is not simple task. A previous study by Molnar [28] found that most adblockers use a community-maintained filter list containing a list of CSS selectors that can locate the cookie dialogs.

I will be adopting a similar method to achieve cookie dialog identification and simple interactions

3.2.5.1 Parsing filter list

The community-maintained EasyList[5] Cookie-List contains not only CSS selectors, so it will need to be parsed before it can be used.

There are two sections we wanted from this list. The ”General element hiding rules” and ”Specific element hiding rules”.

The ”General element hiding rules” are CSS selectors for any website that potentially contains a cookie dialog. An example selector from this section looks like this **###consentPopup** which is not the correct format for CSS selectors. Two # need to be deleted before Selenium can use it.

The ”Specific element hiding rules” provide a URL and its specific cookie dialog locating the CSS selector. An example selector from this section looks like this **telkom.co.za###PrivacyNotice**, I will need to separated the URL and selectors and stored them separately.

3.2.5.2 Implementation assumption

Before implementing the cookie dialog identification feature, there are some assumptions I made.

- All websites can have a cookie dialog
- Every website can have only one dialog
- If a website did not show any cookie dialog 30 seconds after onload is fired, then it does not have a cookie dialog

[5] https://easylist.to/

The above assumption is obviously not true. Firstly, not all websites can have cookie dialog since GPDR only requires the cookies dialog for websites that have EU and EAA region visitors, therefore Chinese websites(which are common in the top 1000) rarely have cookie dialogs. Excluding website that is unlikely to have cookie dialogs can save a lot of running time.

Secondly, some websites have a weird design that somehow displays two cookie dialogs while only one is a valid dialog, assuming there is always only one dialog can cause a system error.

Finally, some website does not show cookie dialog after onload event, it will wait until other user consent is made (e.g. Adult content website that is asking for age confirmation). The above assumptions are only made to simplify the implementation process as it can be finished within limited time.

3.2.5.3 Locating cookie dialog

After successfully parsing the EasyList Cookie-list, the cookie dialog locating implementation is trivial. As described in 3.2.2.2 Selenium supports finding elment by CSS selector using **find_element_by_css_selector**.

For each website visited, the system first checks if the current website is in the ”Specific element hiding rules”. If it is, then use the CSS selector specific for this website. If no element is found or the website does not have a specific CSS selector, then try the ”General element hiding rules” CSS selectors exhaustively. Once an element is located with a selector, store the selector and its associated URL to a dictionary, take a screenshot of the potential cookie dialog and continue to the next website. This logic ensures that CSS selectors tried are as little as possible to reduce the running time(Looping through a complete list of CSS selectors takes a significantly longer time).

Occasionally, a dialog may be hidden in the iframe of the website, all elements within the iframe are not directly accessible by Selenium, therefore, the tool will also have to identify all iframe elements, and switch into these iframes to try to locate the cookie dialog using the CSS selectors.

3.2.5.4 Manual Validation

Using EasyList Cookie-list does not guarantee 100% accuracy in locating cookie dialog. There are times where it cannot find the dialog or it found a wrong element.

Therefore, manual validation is required.

I created a simple UI using PyQt5[6] Figure 3.6, it loops through every screenshot of the located elements and a human agent can verify whether the located element is a cookie dialog.

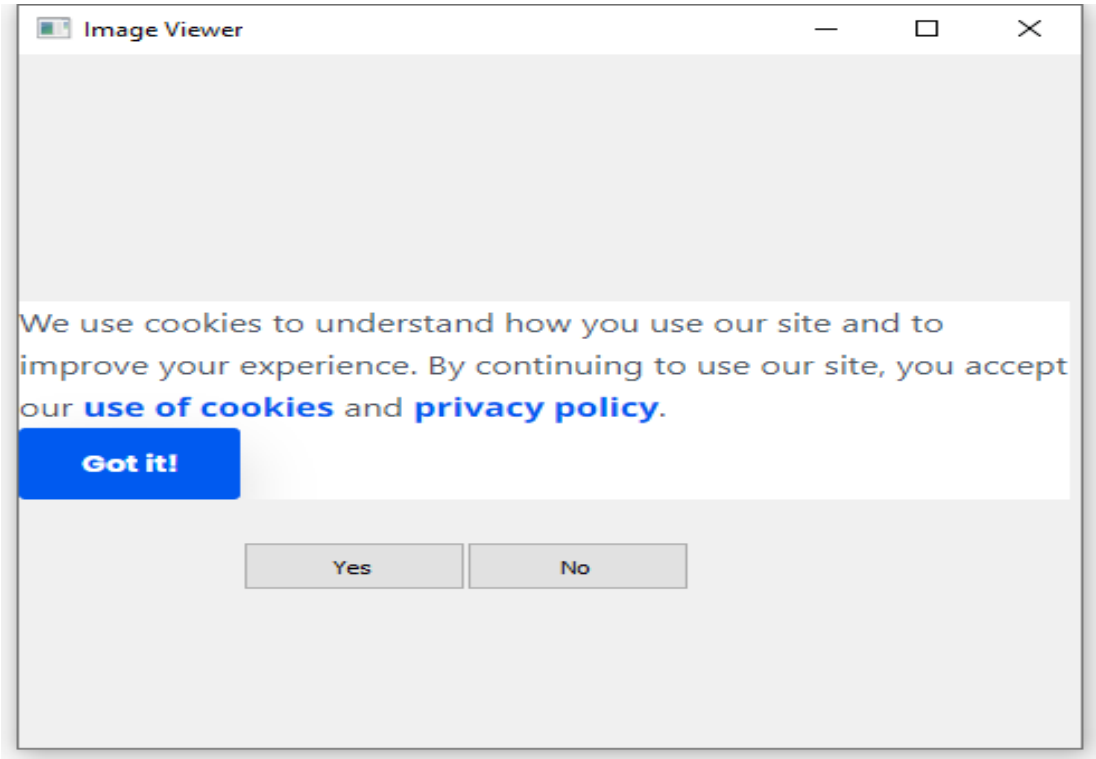

Figure 3.6: Manual validation interface

3.2.5.5 Locating Opt-in option

The websites that system failed to correctly identify a cookie dialog are stored locally, and system is then configured to crawl through these website and I will manually collect the correct CSS selectors using Chrome DevTool, a similar approach as in previous sections 3.2.

After manually validating CSS selectors, I need to locate clickable elements within the cookie dialogs to achieve dialog interaction. As I mainly wanted to investigate whether opt-in cookies with adblockers has any effects on cookies installed, I will be focusing on locating only the buttons for opting-in.

As shown in Molnar's work [28], clickable buttons in cookie dialogs are usually implemented with the following HTML tags.

[6] https://pypi.org/project/PyQt5/

- <button >
- <a >
- tag with attribute role="button"
- <input type="submit" >

Selenium can locate any descendant element of the cookie dialog element that matches one of the above tags to find clickable buttons. This behaviour can be simplely achieved using XPath descendent axes as shown in Figure 3.7

Figure 3.7: Xpath method usage

To identify opt-in buttons from all buttons on the cookie dialog, I have used a list of keywords identify by Monlar [28] that are usually used for opt-in buttons. The list of buttons is checked against this list of keywords, and a match indicating the button element is likely an opt-in button. This list of keywords is only available in English, but there are many websites of other European languages Figure 3.8, to allow Selenium to locate buttons in other European languages, I have used Google Translate API[7] to translate text into English, This ensures "Tout accepter" is viewed as "Accept all" by the system and correct matching is given.

After opt-in buttons is located, its text, HTML class/tag and ID are stored in a dictionary using website domain as key for faster identification in the future, assuming it does not change in the near future.

[7] https://pypi.org/project/googletrans/

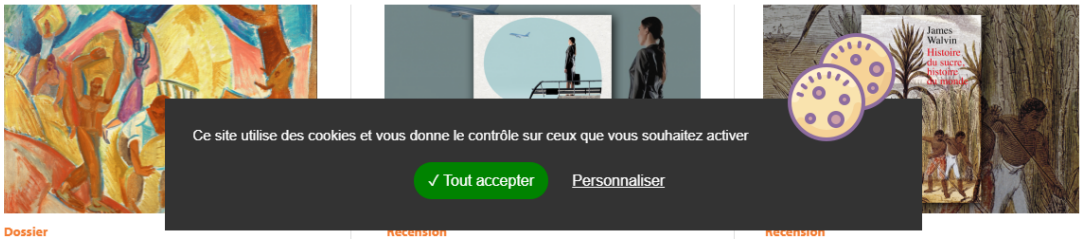

Figure 3.8: Cookie dialog in French

3.2.6 Cookies Distinguishing

As described in Section2.2, there are different types of First-Party and Third-Party cookies, and here I will explain how I distinguished them programmatically.

3.2.6.1 Session and Persistent

The key difference between session and persistent cookies is the presence of the ”Expire” attribute. In a real-world implementation, websites do not always follow the rules, session cookies can still include the ”Expire” attribute. Instead, Website creators sometimes use attribute values of 1 or -1 to represent session cookies. Furthermore, recall that cookies use UNIX time(integer)to represent DateTime, Therefore, to distinguish between session and persistent cookies, I can loop through all First-Party cookies and check if ”Expire” attribute is present(if not present, then it can only be session cookie), and check if the attribute value is greater than 10(to deal with unexpected cookie design).

3.2.6.2 ID-like

Recall Section 2.1 , ID-like cookies are cookies that have some form of ID-like string and is long-lived.

Many previous studies proposed different methods of identifying ID-like cookies.In the research by Englehardt et al. [31][13], they define a cookie to be an identifier cookie if it is long-live(expire time over 3 months) and used the Ratcliff-Obershelp algorithm to compute a similarity score between all cookies value strings, the cookies with a value string of low similarity are considered ID-like cookies.

Another research by Sanchez-Rola et al.[12] took inspiration from other studies on passwords and decided to use a password strength measuring tool zxcvb on cookie string values. This password strength is defined as the amount of information it carries,

since ID-like cookies are required to identify one particular user out of billions of internet users, the more information it carries, the more likely it can identify an individual hence ID-like cookies will have a high password strength.

My implementation will be based on the above methods proposed. Firstly, I will first filter out all cookies with an expiration time of fewer than 90 days for all third-party cookies, as these are too transient for user tracking over time.

Secondly, because many websites use a single cookie's value string to include multiple key-value pairs, I will adopt the assumption proposed in research that each cookie value string can have the following format.

name1=value1 & name2=value2 & name3=value3

Where the pair separating characters represent any characters except "a-zA-Z0-9_-=" [13], to achieve this, I can use the regular expression [ˆa-zA-Z0-9_=-] to split the cookie string into a list of key-value pairs.

Finally, I will be using the same criteria as [12], for all key-value pairs of a cookie, if any value is getting a zxcvb score of 3 or above, the cookie is considered ID-like cookies. This score indicates that the string will take more than 10^9guesses in an offline slow-hash scenario[32] and can be used to distinguish at least 1 billion unique users[12].

As described in Section 2.2.3 First-Party cookies can be ID-like but as it is not shared cross-site, there is often no privacy concerns. Therefore ID-cookie identification is only made for Third-Party cookies.

3.3 System evaluation

The measurement successfully crawled and collected data from 943 websites out of the top 1000. In the 57 websites that failed, 53 websites failed due to various HTTP error codes and 4 website failed due to timeout(onload never fired).

3.3.1 Cookie collection

In my implementation, I have assumed that Chrome DevTool can correctly extract First-Party and Third-Party cookies based on initial observations. To confirm this assumption, I collected cookies from 100 websites sampled from Tranco Top 1000 and generated ground truth cookie types by distinguishing the cookies using a comparison of current URL and cookie domain attribute. This ground truth result is then compared

with cookie distinguishing using Chrome DevTool only.

	No. Site	All	First-Party	Third-Party
Ground truth	100	1356	1059	297
Chrome Devtool	100	1354	1061	293

Table 3.1: Cookie collection comparison

As we can still from table 3.1, both method successfully collected cookies from all 100 websites and the number of cookies collected are almost identical(acceptable margin of error due to website changing constantly) indicating Chrome DevTool is a reliable method of distinguish cookie types.

3.3.2 Cookie dialog and opt-int button identifying

The same approach is applied to evaluate the performance of cookie dialog collection, Table 3.2 . I have again sampled 100 websites from Tranco top 1000, then manually visited these websites and generated the ground truth for the presence of cookie dialogs. Out of 100 websites, 47 had a cookie dialog. In these 47 cookies dialogs, 40 had a button representing the opt-int option, with the remaining 7 being a ”notice dialog” only

My measurement correctly identified 19 dialogs using only the EasyList filter rules. After manual validation and collecting CSS selectors using the Chrome Dev tool, it manages to correctly identify all 47 dialog(given website layout does not change) and 36 of the Opt-in buttons using the list of opt-in keywords.

	No. Site	Cookie dialog	Opt-in buttons
Ground truth	100	47	40
Chrome Devtool	100	Pre-validation(19) Post-validation(47)	36

Table 3.2: Cookie collection comparison

3.3.3 Robustness

Due to time limitations, no extensive tests are done for the robustness of the system. However, comprehensive error catching mechanisms are applied to ensure the system

does not crash prematurely. For this project, the system ran a total of 50 crawl sessions of 1000 websites, only one session failed due to bet365.com. The system is halted indefinitely due website's anti-crawling features. As I could not find a solution to bypass anti-crawling, I have replaced bet be365.com with ibicn.com, which ranked 1001 in Tranco List(06/07/2021) to make up for the top 1000 websites.

3.3.4 Usability

The final version of the measurement tool has poor usability. Different features of the system can only be accessed through command line parameters or manually a couple of lines of code. A GUI is desirable for better usability. However, as this is primarily a prototype and it was sufficient for me to conduct experiments, I have decided to commit my time to other parts of the project. Although, I did implement the session restoring feature to help myself in case of losing all progress due to a power/network outage

Chapter 4

Experiments

In this chapter, I will be outlining the configuration detail used for each experiment conducted

4.1 Defining protection levels

The adblockers investigated in this project does not have clear division between different protection levels they provide. Therefore I have investigated the similarity between all protection options and defined three protection levels for all adblockers.

I have found following protection options to be shared(with small different in wording) among Adblock Plus and Adblock

1. Allow all acceptable(non-intrusive) ads
2. Only allow ads without third-party tracking
3. push notification
4. media icons tracking

AdBlock Plus has uniuqe a protection option named ”Block Additional tracking” with an description saying ”Protect your privacy from known entities that may track your online activity across websites you visit”. This is a very gerneric exaplaning and does not tell what is actual happening which this option enabled

Adblock also has a unique protection option named ”EasyPrivacy”, which enables a set of optional filter rule from EasyList, it claim to ”completely removes all forms of tracking from the internet, including web bugs, tracking scripts and information collectors, thereby protecting your personal data”

Ghostery is slightly different as it come with Free and Paid version. For the free version of software, User is only provided with three protection option.

1. Block default: Advertising, Site Analytics, and Adult Advertising trackers blocked
2. Block everything: All trackers blocked(Advertising, Site Analytics, Adult Advertising, Customer Interaction, Social Media, Essential, Audio/Video Player, Comments, CDN and Misc)
3. Block Nothing: No tracker blocked

Knowing the similarity and differences of all protection options, I have defined three different protection levels, Default (Default installed setting), Medium and High. Each protection level have its own set of protection options enabled shown in Figure 4.1.

Setting / Method	Default	Medium	High
AdBlock Plus	Default installed setting (1)	Default + 2,3,4,5	Medium + Block Additional tracking
AdBlock	Default installed setting (1)	Default + 2,3,4,5	Medium + EasyPrivacy
Ghostery	Default installed setting (Block default)	N/A	Block everything
Mass opt-out (DAA/EDAA)	On	N/A	N/A

Note:
1. Allow all acceptable(non-intrusive) ads
2. Only allow ads without third-party tracking
3. Block push notification
4. Block media icons tracking
5. Chrome Do-Not-Track

Figure 4.1: Protection level definition

Enabling “Only allow ads without third-party tracking” For AdBlock Plus and AdBlock will prompt a message requiring user to enable Do-Not-Track, therefore DNT is enabled for both adblockers at medium and high protection

4.2 Experiment constant variable

There are three constant variable, each can greatly affect the outcome of the experiment.

1. Browser: Chrome version 91.0.4472.101 (09/06/2021)

2. Tranco List(downloaded on 06/07/2021)

3. Selenium 4

The version of Chrome is kept constant, as each version can have a slightly different privacy policy. For example, Google announced that Chrome would "block third-party cookies until late 2023."

This Tranco List is also kept constant, ensuring that the same domains are analyzed among experiments, although this list only changes by about 0.6% of the domains daily.

The version of Selenium affect the functionalities of the entire system i.e. Selenium 3 does not have native support for Chrome DevTool.

4.3 Experiment setups

4.3.1 Baseline

The baseline configuration is simple, No Chrome options or Plugins are enabled. A clean install of Chrome that crawl through the top 1000 website with cookie clearing in-between. This experiment will provide a baseline for all other experiment to compare with.

4.3.2 Mass opt-out DAA/EDAA

There are two experiments for mass cookie opt-out.

1. Experiment 1: opt-out all Ads using DAA(AdChoices)

2. Experiment 2: opt-out all Ads using EDAA(YourOnlineChoices)

Recall that some Ad providers failed to respond to opt-out requests. Therefore, for each experiment, opt-out procedure is repeated 3 times

4.3.3 Adblocker

The Adblockers experiment are conducted according to the protection level table, one experiment per each protection level. So, AdBlocker Plus and AdBlocker will both have three experiments(Default, Medium and High), and Ghostery will have only two(Default, Medium)

4.3.4 Realistic browsing simulation

The realistic browsing simulation refers to having Adblockers enabled but still click opt-in buttons to get rid of annoying cookie dialogs, which is likely happening for many Adblocker users. Due to time limitation, this experiment is only conducted once for each AdBlocker at High protection level.

Chapter 5

Result analysis

In this chapter, I will be showing the result of all experiments and the interpretation of the collected data.

This Tranco top 1000 experiments were all conducted between 29/07-03/08 where 912 websites were successfully collected.

5.1 Abbreviation used

The adblocker abbreviation will follow by **M(Medium protection)** or **H(Max protection)**

1. DAA - Digital Advertising Alliance
2. EDAA - European Interactive Digital Advertising Alliance
3. AdB - AdBlock
4. AdBP - AdBlock Plus
5. GH - Ghostery

5.2 First-Party cookies

To compare the effect of different adblockers on First-Party cookies, I have counted the number of websites that set First-Party cookies. Figure 5.1 shows out of 912 websites. The baseline experiment had 834 websites setting First-Party cookies.

AdB and AdBP had a negligible effect on First-Party cookies at the default setting, with 834 and 832 websites, respectively. This is an expected result as both at default

only have “Allow Acceptable Ads” enabled, and since the First-Party cookie is not used for interest-based advertising, it should not be the target for these adblockers.

Ghostery, However, significantly decrease the number of website with any First-Party cookies. The default setting of Ghostery states, ”Advertising, Site Analytics, and Adult Advertising trackers blocked”. Since some First-Party cookies are used for analytics, this could be the reason that Ghostery blocked much more First-Party cookies than others.

Ghostery at Highest protection blocks 7 more categories of trackers but only reduced 9 more websites, this further confirms my statement above, that analytic First-Party cookies are being blocked as it is unlikely that First-Party cookie are playing a role in general advertising or adult advertising.

At the medium protection level for AdB and AdBP, the number of website with any First-Party cookies still remained roughly the same as the baseline. This is expected now since I know that the additional protection option defined at medium level are intended to block analytic cookies.

Only at the highest protection level, AdBlock and AdBlock Plus blocked significantly more First-Party cookies and arrived at about the same level as Ghostery. This means AdBlock plus’ “Block additional tracking” and AdBlock’s EasyPrivacy can block analytic First-Party cookies

There were no surprises that DAA and EDAA also had a negligible effect on First-Party cookies as there are also primarily used to opt-out tracking cookies and have no effect on First-Party cookies in general

From Figure 5.2 we can also conclude that analytic cookis are not only Persistent cookies, some session cookies are also problem used for website analytic since there are significant drop in both session and persistent cookies when Ghostery(default) is enabled.

The previous two graphs only looked at the number of websites. A more detailed statistical information of First-Party cookies are presented in table 5.1. Although the number of website with First-Party cookies stayed roughly the same for DAA,EDAA and two adblockers at default, There is still some reduction in the average number of First-Party cookies per website, indicating that although they did not completely remove First-Party cookies like Ghostery, they have still removed some First-Party cookies.

To make sure that there is a statistically significant difference, I have conducted a t-test between each browser configuration and the baseline, it turns out that all config-

uration except EDAA is statistically significant with a p-value<0.05. This is probably related to number of participanting advertisers that EDAA had, which was less than DAA, and I have observed 280 opt-out cookie set by DAA, but only 190 opt-out cookies set EDAA.

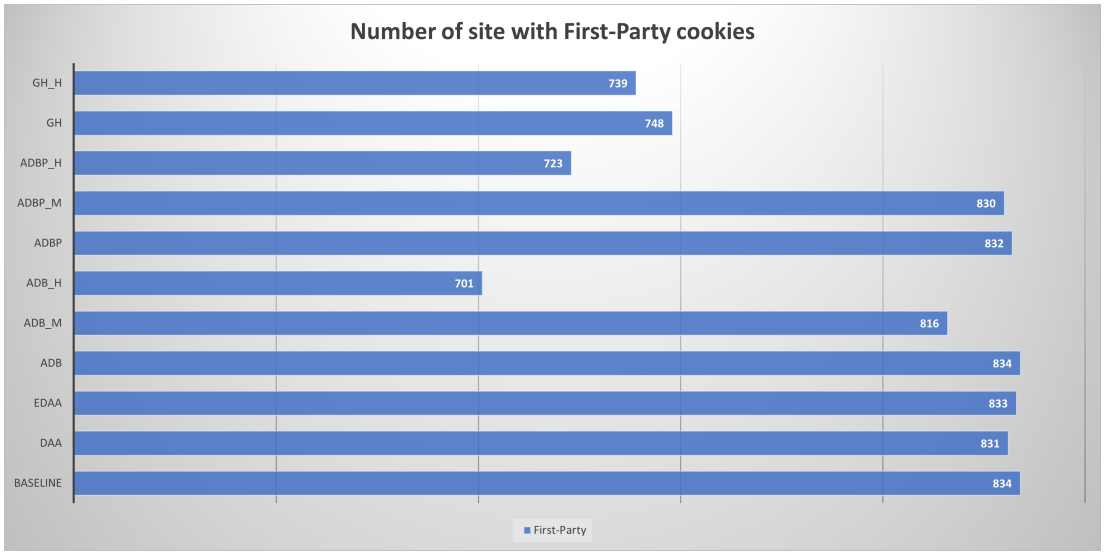

Figure 5.1: Number of sites with at least one First-Party cookie

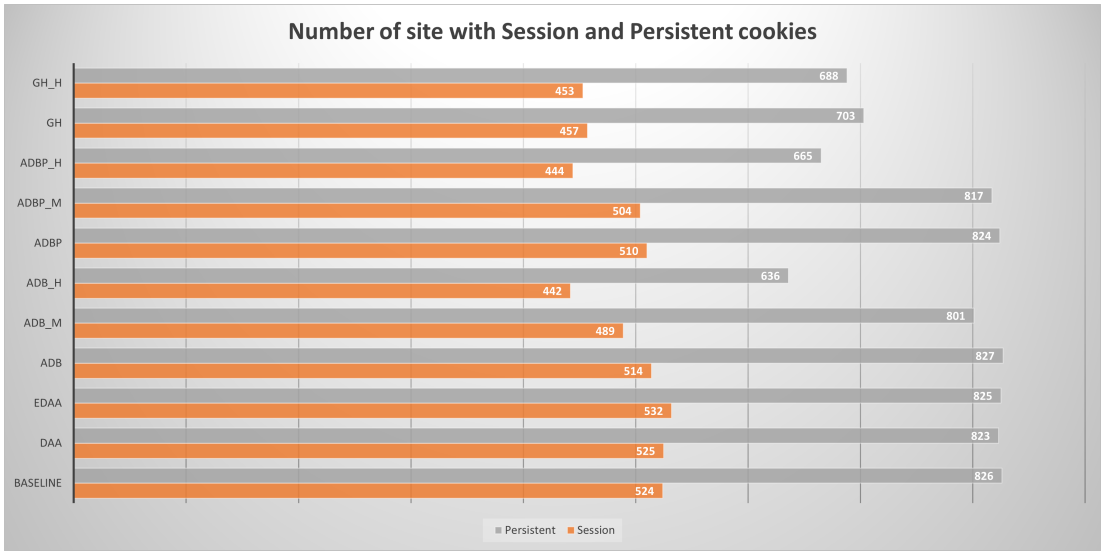

Figure 5.2: Number of sites with at least one session/persistent cookie

Overall, at default setting(most user), Ghostery is best-performing tool in blocking First-Party cookies, at Max protection setting, AdBlock is the best-performaing tool followed by AdBlock Plus. DAA had a little effect and EDAA had no significant effect.

First-Party	mean	median	mode	Max	std
Baseline	12.3	8	0	89	13.3
DAA	11.1	8	0	65	10.6
EDAA	11.8	8	0	62	11.9
AdBlock	10.8	8	0	42	10.5
AdBlock(Mid)	9.4	7	0	67	9.5
AdBlock(High)	4.3	3	0	63	5.3
AdBlock Plus	10.9	8	0	42	10.7
AdBlock Plus(Mid)	10.2	7	0	51	10.0
AdBlock Plus(High)	4.7	3	0	50	5.4
Ghostery	5.6	3	0	61	6.8
Ghostery(High)	5.4	3	0	67	6.7

Table 5.1: First-Party cookie statistics

5.3 Third-Party cookies

Similar analysis methods are used to compare the effectiveness of different tools on blocking Third-Party cookies.

From Figure 5.3, we can see that out of 912 websites. The baseline experiment had 540 websites setting Third-Party cookies.

A very similar pattern as First-Party cookies is observed. At the default setting, AdB and AdBP completely eliminated the Third-Party cookie from 4% of the websites. This may be an indication that that most of the websites only had acceptable Ads. However, if we look at Figure 5.4, we can see they still only eliminated ID-like cookies from 5% of the website. This means that the “acceptable Ads”that they claimed is probably not so acceptable.

Even at the medium setting where “ Only allow ads without third-party tracking” is enabled, AdB and AdBP only further eliminate Third-Party cookies from 10% and 7% of the website, respectively.

This result can mean two thing. It is either that AdB and AdBP is not doing what it claimed to do or in this Post-GDPR time, most websites are using acceptable ads with only the exception of domains from other regions, e.g. China, that are showing up in our results.

At max protection, AdB and AdBP finally made some significant impact, with

both eliminating Third-Party cookies from 56% of websites. This impact is bigger if we look at ID-like cookies, where they eliminated ID-like cookies from 70% of the websites. Since their performance is almost identical, it is safe to say that both AdB and AdBP are using the same filter list, and AdBP's "Block additional tracking" probably enables the EasyPrivacy List like AdB

Once again, Ghostery at default is performing really well by eliminating Third-Party cookies from 65% and ID-like cookies from 62% of the websites. Ghostery is only doing slightly better at max protection, meaning that the other 7 categories are not the primary source of user tracking cookies.

Although DAA and EDAA claimed they help users opt-out advertising cookies, the result shows a different story. DAA performs on the same level as AdB and AdBP at a medium setting, and EDAA performs on the same level as AdB and AdBP at the default setting. Again, considering there are many Chinese websites in the top 1000, this could mean many global websites are complying with the GDPR regulation, and the data we are seeing is mainly due to the presence of foreign websites.

A more detailed statistical information of Third-Party cookies is presented in table 5.2. Again, it has a similar pattern to the First-Party cookie. Even though DAA, EDAA, AdB(default) and AdBP (default) did not completely eliminate all Third-party cookies, they still eliminated some of them.

I have again conducted a t-test between each browser configuration and the baseline. This time every configuration is statistically significant with a p-value$<$0.05 when comparing with baseline, which is somewhat obvious since the differences in means are large enough. This shows that having any kind of cookie-blocking tool installed can be helpful if a user wishes not to be tracked online.

Overall, at default Ghostery is still the best-performing tool. Even at max setting, AdB and AdBP only slightly outperform Ghostery. The performance of DAA/EDAA is visible but not worth the effort of going though the annoying opt-out process.

5.4 Webpage elements

We have seen what different tools can do in terms of cookie blocking. In this section, we will be comparing the effect of these tools on the webpage. To draw a comparison, I have collected the number of HTML elements and the number of JavaScript elements present on the TOP 1000 websites.

From table 5.3, we can see that at the default setting, there are obvious patterns in

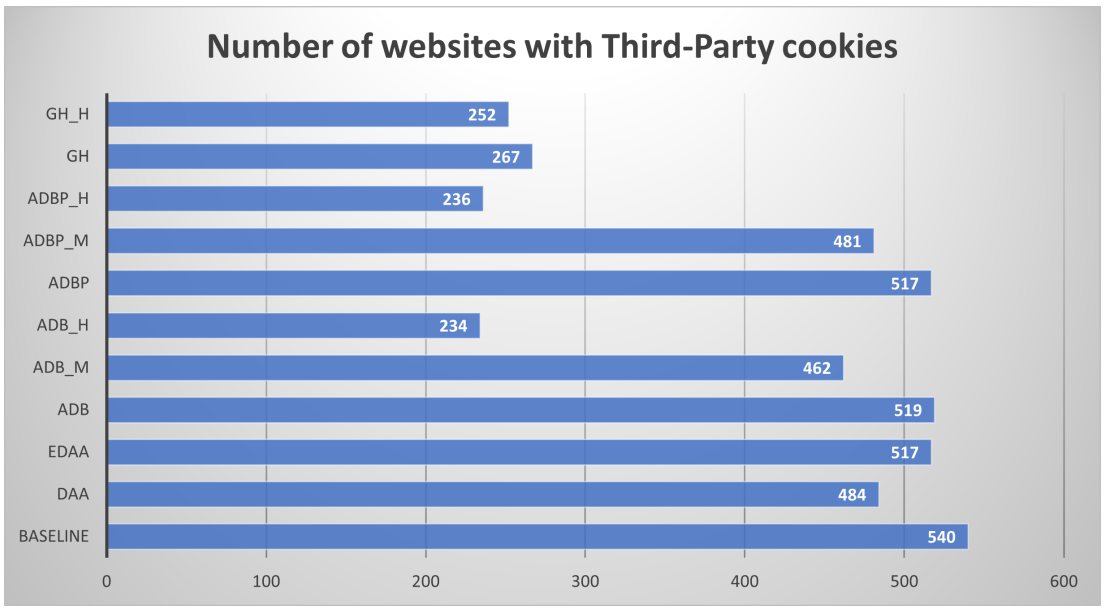

Figure 5.3: Number of sites with at least one Third-Party cookie

terms of the general HTML elements. Although we can see that for AdB and AdBP, as the protection level increase, there is a slight drop in the means, but it is not significant.

To verify this, I performed t-tests between each configuration and the baseline and found all p-value>0.05, meaning that none of the browser configurations was statistically significant. I have also performed t-tests within each Adblockers, and the differences between different protection configurations were not statistically significant.

The result indicates that overall, these tools have almost no effect on the webpage contents. However, many Third-Party cookies are set by JavaScript elements. It is possible that JavaScripts are more affected by these tools.

From table 5.4, we can observe a similar pattern, as the protection level increase, the mean of JavaScripts elements decreases. This time the mean differences are more significant than general HTML elements. Therefore, I performed t-tests once again between baseline and each configuration. The result shows that AdB and AdBP at max protection and Ghostery(default and max)had p-vaule<0.0005, meaning that statistically, there are significant differences when comparing with the baseline.

Overall, this means having Adlockers at higher protection may affect the functionality of a website since it may significantly decrease the number of JavaScript presence on a website, which usually plays a vital role in webpage interaction.

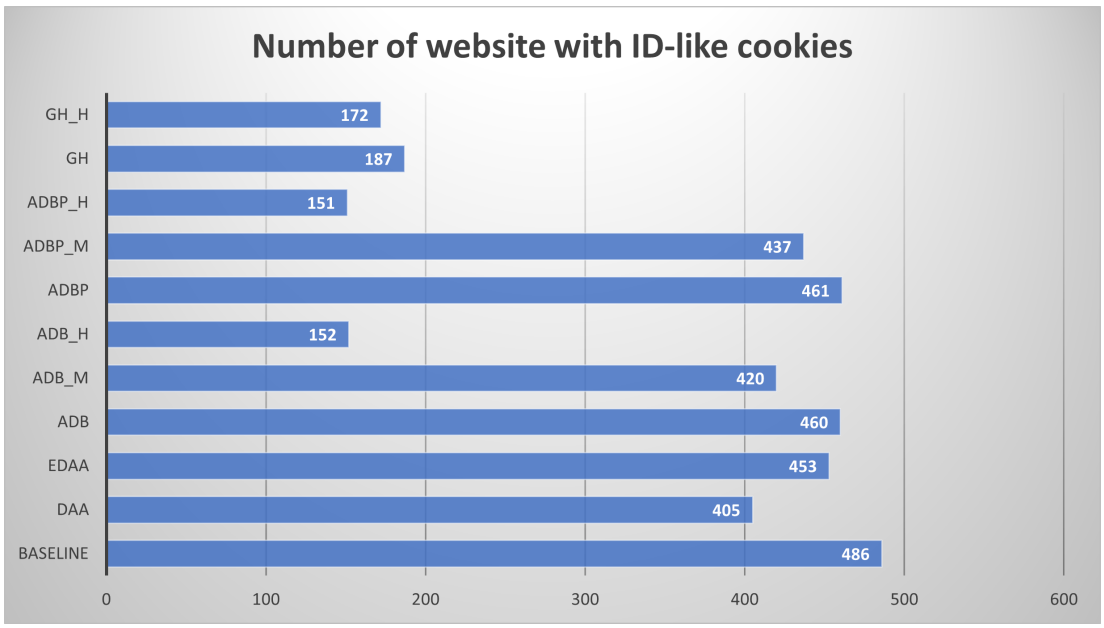

Figure 5.4: Number of sites with at least one ID-like cookie

5.5 Adblockers with opt-in

The realistic browsing simulation experiments did not complete as expected. From the initial observation of data, I can see that at least at the max protection level for each adblockers, whether clicking the opt-in button does not make any significant difference.

I have manually checked what happened in the background using the Chrome DevTool Network panel, and it seems that when a user clicks on the opt-in button, another HTTP GET request is sent. This may be the reason what opt-in buttons had no effect when max protection adblockers are enabled. Recall that adblockers works by filtering HTTP requests. If the opt-in button only makes another HTTP request, it is likely it gets filtered. However, it is possible that interacting with cookie dialogs can have some significant effect when adblockers are at default protection since default does not really offer much protection, as shown by the previous section.

	mean	median	mode	Max	std
Baseline	6.5	1	0	102	12.9
DAA	2.9	1	0	51	5.8
EDAA	4.6	1	0	73	8.7
AdBlock	4.3	1	0	59	7.9
AdBlock(Mid)	2.4	1	0	34	4.3
AdBlock(High)	0.7	0	0	11	1.6
AdBlock Plus	4.3	1	0	69	8.2
AdBlock Plus(Mid)	2.6	1	0	33	4.5
AdBlock Plus(High)	0.7	0	0	11	1.6
Ghostery	1.0	0	0	29	2.5
Ghostery(High)	0.9	0	0	29	2.4

Table 5.2: Third-Party cookie statistics

	mean	median	mode	Max	std
Baseline	1440	1017	324	23645	1819
DAA	1440	1027	312	23839	1811
EDAA	1436	1032	330	23916	1794
AdBlock	1435	1017	324	23641	1817
AdBlock(Mid)	1374	965	324	23648	1795
AdBlock(High)	1368	946	324	35864	1989
AdBlock Plus	1429	1018	324	23851	1814
AdBlock Plus(Mid)	1428	1017	324	23844	1811
AdBlock Plus(High)	1409	994	324	23706	1805
Ghostery	1366	977	325	20581	1632
Ghostery(High)	1369	978	313	20627	1634

Table 5.3: Webpage HTML elements statistics

	mean	median	mode	Max	std
Baseline	39.5	31	15	896	44.7
DAA	38.8	30	8	848	43.1
EDAA	39.3	30	15	926	44.9
AdBlock	38.7	30	15	896	44.1
AdBlock(Mid)	37.1	28	15	896	43.9
AdBlock(High)	31.8	23	15	888	41.3
AdBlock Plus	38.9	31	15	856	43.5
AdBlock Plus(Mid)	38.2	30	15	856	43.2
AdBlock Plus(High)	32.2	23	15	848	40.4
Ghostery	32.5	24	15	860	41.3
Ghostery(High)	31.5	24	8	840	40.7

Table 5.4: Webpage JavaScript elements statistics

Chapter 6

Future Work

6.1 Cookie synchronization

Another research direction yet explored is Cookie synchronization. Cookie sync happens when two different advertising companies share the unique ID of users they collected, this process bypasses the Same-Origin Policy. By detecting cookie synchronization, we can build a visualization of an ecosystem of online advertising. It would be interesting to see how Adblockers at different protection levels interfere with the ecosystem and whether these adblockers favour bigger companies.

6.2 Cookie dialog interaction

Due to time limitations, I was unable to conduct enough experiments for cookie dialogs. Firstly, cookie dialogue opt-in experiments should be conducted for all protection levels(especially default) of adblockers to investigate whether it is safe for a user to click opt-in when they have adblockers enabled. Secondly, many cookie dialogs are not aware of cookies installed on a browser. It would be interesting to see the behaviour of cookies if I opt-in cookies after I have opt-out all cookies using DAA/EDAA. Finally, cookie dialogs now support setting changes where one can decide to opt-in only ”Strictly necessary cookies.” but opt-out others. The First-party cookie experiment shows that some adblockers(default) were not able to block analytic cookies. If we run expereiment to only opt-in for ”Strictly necessary cookies”, we can further confirm whether the finding I had was true. However, this setting change requires more advanced cookie dialog interaction, which my tool currently does not support.

Chapter 7

Conclusions

To conclude, I have successfully met my three primary goals. Firstly, I have successfully designed and implemented a cookie measurement tool that can automate the setup of various browser configurations and accurately collect and distinguish different cookie types. As an additional implementation, the tool can also identify cookie dialogue and locate the opt-in buttons on the identified dialogs. Secondly, I have used the implemented tool to conduct various experiments trying to evaluate and compare the effectiveness of various cookie-blocking tools. Finally, through analysing the collected data, I have found the best-performing cookie-blocking for different usage scenarios.

7.1 Recommendation

7.1.1 To users

For an average user who knows nothing about online tracking, cookies and how adblockers works, but you want a tool that protects your online privacy. The safest cookie option is Ghostery. Out of all configurations investigated, Ghostery provided the best protection at the default setting. If you are willing to play around with adblockers and filter rules, then picking either Adblock or Adblock Plus offers more customization and can further increase the protection performance if you max out all protection options and use custom filter lists. At the current state, do not bother with DAA/EDAA cookie opt-out, they inconvenient to use and requiring much more effort while offering little protection.

7.1.2 To Policy makers

It is now three years since GDPR is enforced, and as a European web user, over half of the websites in the top 1000 still sets Third-Party cookies before me giving any form of consent, which is clearly a violation of the cookie compliance specified by GDPR. However, the good news is that it does seem that more websites are complying since the Third-party cookies are only one-third of all cookies collected compared to two-third of all cookies shown in other studies.

DAA/EDAA was launched so advertising companies can remain self-regulate. The data from this study suggest that the solution they provide offer so little protection that a user is probably better off using any adblockers. The way DAA/EDAA works, if an advertising company wish not to be included in the opt-out process, they can simply stop responding to all opt-out request. While conducting the experiments, I have found at least 8 of DAA participating advertising companies always fail for opt-out requests.

Bibliography

[1] Hannah Ritchie Max Roser and Esteban Ortiz-Ospina. Internet. *Our World in Data*, 2015. https://ourworldindata.org/internet.

[2] David Lancefield, Mark Ambler, Michael Rauber, and Roshan Patel. Department for culture, media & sport research into consumer understanding and management of internet cookies and the potential impact of the eu electronic communications framework report, 2011. `https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/77641/PwC_Internet_Cookies_final.pdf`.

[3] Brian Dean. Ad blockers usage and demographic statistics in 2021, 03 2021. `https://backlinko.com/ad-blockers-users` accessed 01 Aug 2021.

[4] Digitaladvertisingalliance.org. `https://digitaladvertisingalliance.org/`, accessed 01 Aug 2021.

[5] European interactive digital advertising alliance. `https://edaa.eu/`, accessed 01 Aug 2021.

[6] A research-oriented top sites ranking hardened against manipulation - tranco. `https://tranco-list.eu/`, accessed 01 Aug 2021.

[7] Aaron Cahn, Scott Alfeld, Paul Barford, and S. Muthukrishnan. An empirical study of web cookies. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 891–901, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee.

[8] Set-cookie - http — mdn. `https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie`, accessed 02 Aug 2021.

[9] Richie Koch. Cookies, the gdpr, and the eprivacy directive - gdpr.eu, 05 2019. `https://gdpr.eu/cookies/`, accessed 02 Aug 2021.

[10] All about computer cookies - session cookies, persistent cookies,how to enable/disable/manage cookies, 2015. `https://www.allaboutcookies.org/`, accessed 02 Aug 2021.

[11] What are cookies and similar technologies?, 07 2020. `https://ico.org.uk/for-organisations/guide-to-pecr/guidance-on-the-use-of-cookies-and-similar-technologies/what-are-cookies-and-similar-technologies/`, accessed 05 Aug 2021.

[12] Iskander Sanchez-Rola, Matteo Dell'Amico, Platon Kotzias, Davide Balzarotti, Leyla Bilge, Pierre-Antoine Vervier, and Igor Santos. Can i opt out yet? gdpr and the global illusion of cookie control. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, Asia CCS '19, page 340–351, New York, NY, USA, 2019. Association for Computing Machinery.

[13] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1388–1401, New York, NY, USA, 2016. Association for Computing Machinery.

[14] Balachander Krishnamurthy and Craig Wills. Privacy diffusion on the web: A longitudinal perspective. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, page 541–550, New York, NY, USA, 2009. Association for Computing Machinery.

[15] Xuehui Hu and Nishanth Sastry. What a tangled web we weave: Understanding the interconnectedness of the third party cookie ecosystem. In *12th ACM Conference on Web Science*, WebSci '20, page 76–85, New York, NY, USA, 2020. Association for Computing Machinery.

[16] Martino Trevisan, Stefano Traverso, Eleonora Bassi, and Marco Mellia. 4 years of eu cookie law: Results and lessons learned. *Proceedings on Privacy Enhancing Technologies*, 2019(2):126–145, 2019.

[17] Adrian Dabrowski, Georg Merzdovnik, Johanna Ullrich, Gerald Sendera, and Edgar Weippl. Measuring cookies and web privacy in a post-gdpr world. In David Choffnes and Marinho Barcellos, editors, *Passive and Active Measurement*, pages 258–270, Cham, 2019. Springer International Publishing.

[18] Arunesh Mathur, Jessica Vitak, Arvind Narayanan, and Marshini Chetty. Characterizing the use of browser-based blocking extensions to prevent online tracking. In *Proceedings of the Fourteenth USENIX Conference on Usable Privacy and Security*, SOUPS '18, page 103–116, USA, 2018. USENIX Association.

[19] Arthur Gervais, Alexandros Filios, Vincent Lenders, and Srdjan Capkun. Quantifying web adblocker privacy. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *Computer Security – ESORICS 2017*, pages 21–42, Cham, 2017. Springer International Publishing.

[20] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczynski, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. *Proceedings 2019 Network and Distributed System Security Symposium*, 2019.

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

[22] Chrome devtools. `https://www.selenium.dev/documentation/support_packages/chrome_devtools/`, accessed 05 Aug 2021.

[23] Lukas Schulze. How does it work? — chrome extension downloader. `https://chrome-extension-downloader.com/how-does-it-work.php`, accessed 01 Aug 2021.

[24] Richi Jennings and 2021. Chrome to enforce https web protocol (like it or not), 03 2021. `https://securityboulevard.com/2021/03/chrome-to-enforce-https-web-protocol-like-it-or-not//`, accessed 06 Aug 2021.

[25] Getting started — selenium python documentation. `https://selenium-python.readthedocs.io/getting-started.html`, accessed 10 Aug 2021.

[26] Stack Overflow. Can a website detect when you are using selenium with chromedriver? `https://stackoverflow.com/questions/33225947/`

can-a-website-detect-when-you-are-using-selenium-with-chromedriver/41220267, accessed 01 Aug 2021.

[27] Working with cookies. https://www.selenium.dev/documentation/support_packages/working_with_cookies/, accessed 10 Aug 2021.

[28] Barnabas Molnar. Measuring the cookie-setting behaviour of web pages showing privacy warnings. 2020.

[29] Chrome devtools protocol. https://chromedevtools.github.io/devtools-protocol/, accessed 10 Aug 2021.

[30] 5. data structures — python 3.8.3 documentation. https://docs.python.org/3/tutorial/datastructures.html, accessed 10 Aug 2021.

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

[32] Daniel Lowe Wheeler. zxcvbn: Low-budget password strength estimation. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 157–173, Austin, TX, August 2016. USENIX Association.

Appendix A

Statistical data

	mean	median	mode	std
Baseline	1.8	1	0	2.7
DAA	1.8	1	0	2.7
EDAA	1.8	1	0	2.7
AdBlock	1.7	1	0	2.7
AdBlock(Mid)	1.6	1	0	2.6
AdBlock(High)	1.2	0	0	2.0
AdBlock Plus	1.7	1	0	2.7
AdBlock Plus(Mid)	1.7	1	0	2.7
AdBlock Plus(High)	1.2	0	0	2.0
Ghostery	1.4	1	0	2.5
Ghostery(High)	1.4	0	0	2.5

Table A.1: Session cookie statistics

	mean	median	mode	std
Baseline	10.5	7	0	11.9
DAA	9.3	6	0	8.9
EDAA	10.0	6	0	10.5
AdBlock	9.0	6	0	8.8
AdBlock(Mid)	7.8	5	0	7.7
AdBlock(High)	3.1	2	0	3.9
AdBlock Plus	9.2	6	0	8.9
AdBlock Plus(Mid)	8.5	6	0	8.2
AdBlock Plus(High)	3.4	2	0	4.1
Ghostery	4.1	2	0	5.0
Ghostery(High)	4.0	5	0	4.9

Table A.2: Persistent cookie statistics