Cookie Dialog Compliance

Graduation Assignment

Towards automating measurements of cookie dialog compliance

by

Koen Berkhout

Maarten Meyns

Students:	Koen Berkhout	
	Maarten Meyns	
Thesis committee:	Prof. Dr. Tanja Vos,	Open University
	Dr. Ir. Hugo Jonker	Open University
	Dr. Fabian van den Broek	Open University
Course code:	IB9902	

Open Universiteit
de beste! www.ou.nl

CONTENTS

List of Figures

List of Tables

ACRONYMS

API Application Programming Interface. 5, 24

ccTLD Country code top-level domain. 4, 5, 44, 45, 48, 49, 70

CMP Consent Management Platform. 5, 9, 10, 18, 36–38

CNIL Commission Nationale de l'Informatique et des Libertés. 5, 49, 50

CSV comma-separated values. 5, 14, 19, 25

DNS Domain Name Server. 5, 45, 46

DPA data protection authorities. 5, 8, 33

GDPR General Data Protection Regulation. 5, 8, 9, 30

GUI Graphical User Interface. 2, 5, 23

HTML Hypertext Markup Language. 5, 37, 38, 40, 41

HTTP Hypertext Transfer Protocol. 5, 7, 63, 64

HTTPS Hypertext Transfer Protocol Secure. 5, 63–65, 76, 77, 88

JSON JavaScript Object Notation. 5, 25

OSINT Open Source Intelligence. 5, 17

URL Uniform Resource Locator. 5, 88

1 INTRODUCTION

This report belongs to the graduation project that is part of the bachelor program Computer Science at the Open University. In this project, the compliance with cookie preferences is evaluated in both a semi-automated and fully automated way.

CONTEXT

Cookies can store information such as the items in a shopping cart or whether a user account is logged in on a certain device. In recent years, however, cookies have been increasingly used for purposes other than these essential functionalities. Cookies can also be used by websites to collect information about their visitors. Most websites implement consent dialogs to request permission for collecting personal data due to legislation such as the ePrivacy directive. There are other ways to collect this data, for example browser fingerprinting, but this project focuses exclusively on the use of cookies. This is commonly implemented by means of cookie dialogs shown on the first visit to a website. There are various legal grounds based on which cookies may be placed. Although the interpretation of current legislation is inconclusive and still an ongoing topic of discussion among experts, consent is the only legal basis that websites can rely on for placing tracking and marketing cookies. The legal grounds based on which the placement of cookies can be justified are further discussed in Section 2. Some cookie dialogs can be deceiving or overly complicated. They may use so-called dark patterns to persuade a visitor to accept all cookies and give permission to collect their personal information.

CONTRIBUTIONS

Although the presence of cookie dialogs suggests that users can decide which cookies they accept, the possibility exists that websites do not always fully comply with cookie preferences. The goal of this project is to test this assumption in a systematic and automated way. To reach this goal the project is divided into two substudies, each of which is mainly carried out by one student.

The first substudy, conducted by Koen Berkhout, concerns the compliance by websites with the cookie preferences stated by their visitors. The main research question is 'To what extent do websites comply with cookie preferences?'. To this end, the placement of cookies will initially be recorded before any interaction has taken place. From there, preferences are communicated by respectively giving and denying consent. Section 4 outlines the details of this study and the methods for semi-automatically collecting the data. The results are used as a validation set for the web crawler that will be introduced below. This substudy yields information about the placement of cookies and their types, the effort to deny consent measured in clicks, and tools for conducting the experiment.

The second substudy, conducted by Maarten Meyns, concerns the ability to detect cookie dialogs and interact with these cookie dialogs in an automated way. The main research question is 'How can a large list of websites be visited in a controlled manner and cookies and cookie dialogs be documented?'. Section 5 concerns the creation of a fully working crawler that implements this functionality. Machine learning is employed to select the correct cookie dialog and classify the buttons in the cookie dialog, and detects buttons to accept and reject cookies. This substudy yields data on the placement of cookie dialogs and cookies.

2 BACKGROUND

COOKIES

Cookies are a fundamental part of the web since their standardization process started in 1995 [Kri01]. Cookies are an additional layer to the Hypertext Transfer Protocol (HTTP), which by itself is a stateless protocol. When a user requests a webpage, the server sends a response and immediately disconnects. A visit may consist of multiple requests as the user browses through a website. A new connection is initiated with each request, and is completely independent from any previous request. For some applications, however, it is necessary to keep track of state. One example is a shopping cart, in which items should be persisted across multiple requests. Cookies enable web applications to maintain state in the browser by placing a small file in which data specific to a visitor can be stored. The cookies are sent with each response and accessible by the website. Cookies have a set of properties, among which at least a name and a value. On each request the cookie is enclosed in the Cookie header, and the website can read out the value of a cookie by its name. Figure 1 models how cookies are used to transfer data between a browser and a server. Similar behavior can be accomplished by binding state to a user's IP address, as part of the URL, or by using the browser's local storage, but each of them has their disadvantages and limitations which makes them unsuitable alternatives to cookies.

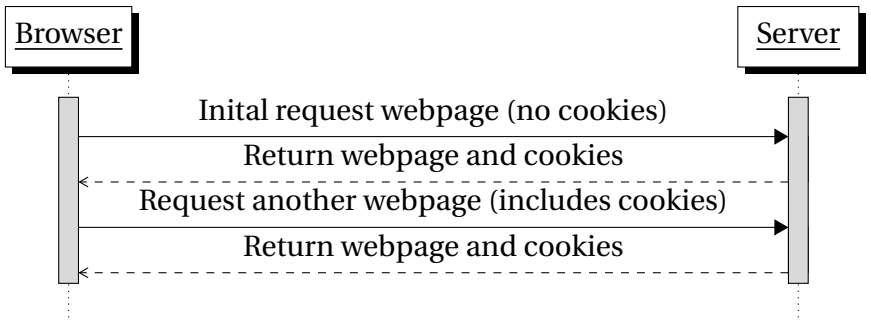


Figure 1: HTTP cookies (adapted from en.wikipedia.org/wiki/HTTP_cookie)

Cookies can be classified by various properties other than their purpose, for example by provenance (first vs. third party) or duration. A detailed categorization by purpose made by Proton Technologies AG distinguishes four types of cookies: strictly necessary, preference, statistics, and marketing.[1] It is not always possible to stringently categorize a cookie in one of these categories, as they may serve multiple purposes. According to this website, funded by the European Union, the greatest privacy risks are presented by third-party persistent marketing cookies. Furthermore, it states that these cookies can contain significant amounts of information about someone's online activity, preferences, and location. Cookies can also be used for tracking visitors across the web. A clear distinction can be made between first- and third-party cookies. First-party cookies are placed by the domain the user is currently visiting, whereas third-party cookies are placed by (or for) other domains. Websites can leverage third-party tracking cookies for mass surveillance [ERE+15], for example by embedding the same tracker on multiple websites. In this way, visits can be linked to unique users, even when these visits come from different IP addresses.

[1] `https://gdpr.eu/cookies/`

LEGISLATION

The 2002 ePrivacy directive, also known as 'the cookie law', was already in force before the GDPR and concerns the online tracking of users and confidentiality of communications.[2] It requires explicit consent for placing cookies, except when strictly necessary. The directive is a guideline along which member states must implement laws and not a binding legislative act. The GDPR, successor of the 1995 Data Protection Directive, was adopted in 2016 and became enforceable in 2018 in the European Union. It is is complementary to the ePrivacy directive, more specifically by defining how data can be lawfully processed. Being a regulation, it is a binding legislative act that generally applies across the EU. Moreover, it gives data protection authorities (DPA) the power to issue large fines to companies who violate the regulation. Hence, it is of no surprise that cookie dialogs have made an upsurge since its entry into force in 2018. The GDPR applies when personal data is processed, and both *personal data* and *processing* are broad concepts in this regulation. "Personal data is any information relating to an identified or identifiable natural person ('data subject'), and processing is any operation or set of operations which is performed on personal data or on sets of personal data, whether or not by automated means" [RGBZ16]. This includes, but is not limited to, collecting, storing, and disclosing personal data. One of the implications of the GDPR is its imposition of rules on third party trackers, which is considered using personal data for 'monitoring' the behavior of users [HvdSB19]. Furthermore, Hoofnagle et al. explain that the processing of personal data requires a legal basis. The six legal bases specified by the GDPR are (1) consent, (2) necessity for a contract, (3) mandated by law, (4) necessity to protect the life of the data subject, (5) public task, or (6) when the interests of the data controller (the company or website) are more important than the interest of the data subject. In most cases, the latter five legal bases do not apply when tracking users of a website, thus leaving *consent* the last resort for website owners. Moreover, the GDPR requires that "consent be freely given, specific, informed, and unambiguous" [HvdSB19]. An issue with this requirement is that by using dark patters websites could persuade users to accept tracking cookies.

DARK PATTERNS

There is no universally accepted definition of what exactly dark patterns are. Nonetheless, in previous papers researchers have come up with descriptions that are useful and usable. Gray et al. define dark patterns as "a construct where designers use their knowledge of human behavior and the desires of end users to implement deceptive functionality that is not in the user's best interest" [GKB+18]. Mathur et al. elaborate on this definition by adding that "this deceptive functionality is used to coerce, steer, or deceive users into making decisions that, if fully informed and capable of selecting alternatives, they would not make" [MAF+19]. Gray et al. define dark patterns based on strategies they employ to trick users into performing an action, one of which is obstruction. In the context of cookie dialogs this could be the effort to make it needlessly complicated to reject cookies, for example by requiring a disproportionate number of clicks to deny consent.

[2] https://gdpr.eu/cookies/

3 Related work

In 2019, Sanchez et al. [SRDK+19] ran an experiment on a selection of 2,000 websites from the Alexa top 1M, in which both the information presented to visitors and the actual cookie tracking behavior is evaluated. The results show that tracking frequently occurs and happens generally without the user's consent, and above all often before or without showing any notification. In approximately 4% of the test cases the website offered an opt-out from tracking. The researchers observe that opting out from tracking is usually not properly implemented, rejecting tracking is ineffective ('false rejections'), and commonly users end up being tracked with long lasting cookies in spite of having denied consent. It is therefore concluded that the spirit of the GDPR is generally not complied with. The partially automated method of data collection is similar to the approach in Section 4 to ensure that cookie dialog interaction is reliably performed. However, this study was conducted three years prior ours, and the findings are compared with the results found in Section 4 to learn whether the situation has changed over time.

Web bots, also referred to as crawlers, are used to automatically visit and interact with websites, thus facilitating web measurements at scale. However, some websites actively try to prevent crawlers from accessing their pages or serve different content upon detection. A study by Jonker et al. [JKV19] found that approximately 13% of the websites of the Alexa Top 1M employ crawler detection techniques. These detections may rely on browser properties and specific objects in the DOM of the browser, also known as the fingerprint surface. Other detection mechanisms target differences in behavior between humans and crawlers, among which keyboard and mouse events. In a comparison of 14 browser automation frameworks, they found that the detection rate varies greatly. The least frequently detected frameworks are WebDriver, Selenium, and Chrome Headless, each of which got detected on approximately 1% of the visited websites. Some of the design choices in Section 5 are partially based on the findings of this study, such as the selection of the automation framework and settings to avoid detection.

The study by Matte et al. [MBS20] conducted in 2020 indicates that many cookie dialogs are implemented by means of Consent Management Platforms (CMPs), which collect and propagate cookie preferences to third parties. In general, it was found that the total amount of third party cookies dropped by 22% after the introduction of the GDPR, and the number of cookie consent dialogs increased. However, the use of CMPs does not guarantee compliance with provided or denied consent. When visiting 22,949 websites with an automated crawl, they found that it is not uncommon for websites to record positive consent before any user interaction has taken place. This data is used to explain the detected presence of cookie dialogs in Section 5.

The graduation project *cookie dialogs and their compliance* [Aer21] conducted by Koen Aerts covers the issue of which kind of cookies are set before consent is given. This is an important starting point, because it allows for the detection of tracking cookies that are set before any interaction has taken place with a user, provided that tracking cookies can be distinguished from non-tracking cookies. To detect the cookies set by websites a crawler was run against Tranco [LPVGT+19], a list compiled of top ranking websites augmented with measures to prevent manipulation, from which a selection of only European websites was made. The study concludes that transparency regarding the purpose of cookies is still below par, and that a considerable number of cookies are set for adver-

tisement and tracking purposes before consent is given. The full scale of the violations is unknown, because the purpose could not be determined for a large part of the collected cookies. The results of this study will be directly compared with the results of Section 5.

In the study *Automating Cookie Consent and GDPR Violation Detection*, Bollinger et al. [BKCB22] find at least one GDPR violation in almost 95% of the analyzed websites. The failure rate is slightly lower in websites that use a Consent Management Platform (CMP), approximately 88%. This study focuses on websites with a CMP, and classifies cookies using decision trees trained with the XGBoost library.[3] To train the classifier, the cookie purposes are collected from consent management platforms, where website owners select the categories in which cookies fall and their purposes. However, these cookie purposes could only be collected from a small set of all CMPs. This study is related to our project, but is mainly complementary given differences in website selection. The failure rate is compared with the results in Section 4.

As part of the study *CookieEnforcer: Automated Cookie Notice Analysis and Enforcement* [KNHF22], Khandelwal et al. developed a tool that can explore a cookie dialog and reject all cookies, even if this option is hidden in a second screen or requires numerous clicks. Although the purpose of the crawler discussed in Section 5 differs from the CookieEnforcer, many of the techniques are used as a building block. This includes, for example, the selection of potential cookie dialogs through z-index values and the use of machine learning to classify text in a cookie dialog candidate.

[3] `https://github.com/dmlc/xgboost`

4 COMPLIANCE WITH COOKIE PREFERENCES

4.1 INTRODUCTION

The presence of cookie dialogs conveys the impression that users can choose which kind of cookies are placed by the websites they visit. Ideally, websites comply with the preferences stated by a user, but in practice it is not clear how effective it is to give or deny consent. This part of the project evaluates the behavior of websites regarding their compliance with cookie preferences. It is important to be aware of the possibility that websites intentionally ignore cookie preferences, but non-compliance could also be unintentional. Creating a website often involves third-party dependencies such as external scripts, and it is difficult for website owners to keep track of their behavior. As introduced before, cookie dialogs may contain various dark patterns to convince a user to provide consent, one of which is obstruction. The extent to which websites comply with cookie preferences is examined in this substudy. In addition, the effort needed to express cookie preferences is measured with the click count metric, which could be an indication of the presence of a dark pattern. Furthermore, a validation set is created for the crawler that will be introduced in Section 5.

4.2 RESEARCH

The research questions below are defined to determine whether websites comply with cookie preferences. Websites comply if they (1) do not place 'unnecessary' (defined in Table 2) cookies at first, and (2) adhere to the cookie preferences stated by a visitor, regardless of intentions. The main research question *To what extent do websites comply with cookie preferences?* is further divided into the following subquestions.

RQ1 WHICH ACTIONS CAN BE PERFORMED IN A COOKIE DIALOG?

Three approaches are considered for determining which actions can be performed in a cookie dialog. The first option is to make use of literature in which cookie dialogs and possible actions are described. However, an initial exploration shows that no such literature is readily available. Secondly, actions that should in theory be possible can be self-invented. This could result in good theoretic examples but may lack practical usefulness. Third, a representative set of websites can be visited to extract the possible action from real-world cookie dialogs. This last approach is preferred, because it is considered time efficient and has the potential to yield useful examples.

Appendix A demonstrates two typical cases. Figure 48 shows the cookie dialog that is displayed when visiting microsoft.com. The 'Accept' and 'Reject' buttons are positioned alongside each other, and there is a third option to manage cookies. Figure 49 shows another case that is regularly encountered. Only the 'Accept all' action can be directly performed, whereas rejecting cookies requires considerable effort. After analyzing various cookie dialogs such as the aforesaid cases, the following actions have been defined:

- INITIAL
  This means doing nothing, i.e. before performing any cookie dialog interaction.
- ACCEPT ALL
  This action can be performed in the presence of an Accept all (or similar) option.

- DENY BASIC
  This action can be performed in the presence of a Deny/Reject (or similar) option.
- DENY ADVANCED
  This action can be performed in the presence of a Manage Cookies (or similar) option, by manually deselecting or objecting to all types of cookies.

RQ2 HOW CAN A SET OF WEBSITES BE VISITED IN A (SEMI-)AUTOMATED WAY TO DETECT WHICH COOKIES ARE PLACED?

Four strategies for visiting websites have been considered to determine which cookies are placed, ranging from manual to completely automated.

1. Manual
Visiting websites: Manual
Dialog interaction: Manual
Inspecting cookies: Manual

In this strategy, all actions are performed manually. Visiting websites by hand from a list is time-consuming. Manual interaction with a cookie dialog is reliable, because there may be many options that are difficult to automatically interpret for a crawler. This task can be more reliably carried out by humans, provided that attention is paid to correctly perform the required action without falling into a dark pattern's trap. Manual cookie inspection and recording is cumbersome.

2. Partially automated: JavaScript in the console
Visiting websites: Manual
Dialog interaction: Manual
Inspecting cookies: Automated

In this strategy, visiting websites and cookie dialog interaction are performed manually. For inspecting cookies, JavaScript is run to extract cookies from the browser. However, cookies may have an http-only flag, in which case the browser prevents the script from retrieving cookies. This leads to incomplete results.

3. Partially automated: browser plugin
Visiting websites: Automated
Dialog interaction: Manual
Inspecting cookies: Automated

In this strategy, only cookie dialog interaction is performed manually. A browser plugin can automatically navigate to the next website on a list, for example by clicking on a 'next'-button. All cookies can be recorded, including cookies with an http-only flag, because a plugin can retrieve all stored cookies by leveraging the browser API.

4. Fully automated: scraper
Visiting websites: Automated
Dialog interaction: Automated
Inspecting cookies: Automated

In this strategy, a web scraper is built to perform all actions automatically. Building a scraper is a project in itself and described in Section 5 of this paper. The results may be less reliable due to detection mechanisms employed by websites or by the misinterpretation of cookie dialogs by the crawler.

Selected strategy
To review the compliance of websites, the partially automated strategy is used by developing a browser plugin that offloads most of the labor-intensive tasks. This strikes a balance between reliability of the results and the time required to develop the tool and conduct the experiment. The plugin has access to a list of websites based on Tranco. Furthermore, it automatically records the cookies set by a website upon the initial visit before any user interaction has taken place. Which cookies are placed can be reliably determined by leveraging the browser API.[4] The reviewer interacts with the cookie dialog by subsequently performing the actions of giving and denying consent as defined in RQ1.

RQ3 How can cookies be classified?

When the placement of cookies by websites is recorded, they need to be classified to determine their purpose. Without proper classification it is not possible to find out whether a website violates legislation, because not all types of cookies require explicit consent. There are various ways in which the purpose of a cookie can be determined.

1. Read the privacy policies and cookie statements of the visited website, and contact the website owner if this provides insufficient data.
2. Make use of external data sources about cookies and their purposes.
3. Estimate the purpose based on properties of the cookie.

The first alternative could be an option for small experiments, but is not feasible for reviewing larger sets of websites. Moreover, websites do not always provide a list of purposes for each cookie they use. Option 2 relies on external data sources, which need to be accurate, have a large sample set, and be publicly available. Option 3 could be used in case there is no information available about a cookie. For example, one could calculate the entropy of the cookie name or value to determine the likeliness of containing a tracking id. However, session ids should also be unique, and this ambiguity makes this option less reliable.

After weighing up the advantages and disadvantages, the decision was made to select option 2. There are three sources that particularly appear when searching for cookie purpose databases.

1. CookiePedia[5] is a website operated by OneTrust and is the "largest database of pre-categorized cookies" containing over 42,000,000 samples.
2. CookieDatabase[6] is a joint project by Complianz and SIDN that continually researches and describes cookies. The database contains approximately 15,500 samples.
3. Open Cookie Database[7] is an open source project on Github that tries to describe and classify the most used cookies. At the time of writing it contains 747 samples.

[4] `https://developer.chrome.com/docs/extensions/reference/cookies/`
[5] `cookiepedia.co.uk`
[6] `cookiedatabase.org`
[7] `github.com/jkwakman/Open-Cookie-Database/`

Neither CookiePedia nor CookieDatabase offer an API or a downloadable database, but appending the cookie name to a GET request on the website leads to a result page including categorization (if any). Browser automation can be used to scrape cookie classifications from these data sources. On the contrary, Open Cookie Database provides a downloadable comma-separated values (CSV) file which can serve as a local database.

When consulting these sources with a small sample set of cookies it was found that sometimes the classifications contradict each other. Additionally, not all cookies are known to all sources and they may thus complement each other. For these reasons all three providers are selected. The following categories of cookies are used: necessary, performance, functionality, and marketing. These categories are linked to the external sources according to the mapping in Table 1.

Category	**CookiePedia**	**CookieDatabase**	**Open Cookie Database**
Necessary	Strictly Necessary	Functional	Functional
Performance	Performance	Statistics/Analytics	Analytics
Functionality	Functionality	Preferences	Preferences
Marketing	Targeting/Advertising	Marketing/Tracking	Marketing

Table 1: Cookie categories mapping

The categories describe cookies with specific characteristics. Table 2 contains descriptions for each of the categories, with an additional entry for the definition of unnecessary cookies. The descriptions are established by Optanon and taken from the CookiePedia purpose descriptions page.[8] Note that only necessary cookies are exempted from explicit user consent.

Category	**Description**
Necessary	"These are all the cookies without which the website could not perform basic functions. They may be set automatically when pages load, or as a result of a user request that cannot be fulfilled without the use of the cookie. Generally these are session cookies that expire on closing the browser but not always. The law allows that any Necessary cookies are exempted from any requirements for user consent."
Performance	"These cookies are used to provide site owners with statistical information about their site – which is generally used for performance measurement and improvement. This is generally also known as 'Analytics'. It includes activities like counting page visits, dwell time, bounce rates, technologies used to access the site, and page load speeds."
Functionality	"These cookies are generally there to support site functionality that is visible or advantageous to the user or their experience of the site. This includes elements of persistent personalisation (remembered on subsequent visits), and enhanced functionality like web chat services, surveys, commenting and rating systems, and user preferences. They are generally a mix of first and third party, session and persistent cookies."

[8] `https://cookiepedia.co.uk/classify-cookies`

Marketing	"These types of cookies are set by digital advertising businesses for the prime or sole purpose of managing the performance of adverts, displaying adverts, and/or building user profiles to determine the display of adverts elsewhere. These will almost always be third party cookies and mostly persistent."
Unnecessary	Performance + functionality + marketing (all except necessary).

Table 2: Cookie category descriptions (by Optanon, taken from cookiepedia.co.uk)

4.3 EXPERIMENT

To determine the level of compliance with cookie preferences, the cookie placement behavior of 400 websites from the Tranco list is examined by three reviewers. The possible actions that have been determined in RQ1 are directly translated to *visiting modes*. A browser plugin ('Cookie Helper') is developed that supports the selection of a mode, loads a website in that mode, and records the cookies and number of clicks after the user has performed the corresponding action (INITIAL, ACCEPT ALL, DENY BASIC, DENY ADVANCED). Each website is visited in all four modes, resulting in a total of 1600 visits. The technical details of the plugin are discussed in Section 4.4.

4.3.1 GOAL

The goal of the experiment is twofold. Firstly, the compliance by websites with the cookie preferences stated by their users is tested. More specifically, the resulting data includes cookie counts for each of the modes segmented by cookie purpose. Analyzing which (types of) cookies are placed in each mode yields important information to base conclusions on. Moreover, the resulting data is used as a validation set for the crawler. Secondly, the number of clicks is recorded as a quantitative measurement of the effort involved with completing an action. Although the focus of this study is not on dark patterns, a major imbalance between accepting and rejecting cookies is an indication of the presence of the *obstruction* pattern. This pattern shall be considered present if the required number of clicks is at least three times higher to reject all cookies than to accept them.

4.3.2 PREREQUISITES

The websites are visited by three different reviewers, and to ensure consistent results the following guidelines are followed.

- The Tranco list generated on June 16 is used.[9]
- All websites are visited within a two-week time span in week 24 and week 25.
- The latest version of the Google Chrome browser is used (at present v103).
- Chrome is reset to the default settings without logged-in user and runs without any other plugins besides the Cookie Helper.
- Chrome is set to allow all cookies, including third-party cookies.

[9] `https://tranco-list.eu/list/Q9XG4`

- In Chrome's security settings, the Secure DNS provider is set to Cloudflare. Using Secure DNS enables Chrome's built-in DNS-over-https resolver, effectively bypassing DNS ad-blocking techniques such as Pi-Hole or AdGuard Home.
- Websites are visited with a direct connection, without a proxy or VPN.

4.3.3 DOMAIN SELECTION

The Tranco list is divided into four 'buckets' according to the composition scheme in Table 3. From each bucket, 100 random websites are visited. This number is limited due to manual nature of this study, but is expected to yield a sufficient amount of data to analyze. The same list of websites will be visited and the same composition of buckets will be reused in Section 5 but expanded upon. A selection with incremental bucket sizes is made based on the assumption that higher-ranked websites are more influential and should therefore be more strongly represented in this study. The first bucket's size is chosen such that there is a sufficient number of websites to review after filtering (see Section 4.3.4), and each subsequent bucket is enlarged with an intuitively chosen variable scale factor to ensure an appropriate representation.

Bucket	Range start	Range end	Scheme
1	1	5,000	Random 100
2	5,001	25,000	Random 100
3	25,001	100,000	Random 100
4	100,001	1,000,000	Random 100

Table 3: Bucket composition scheme

4.3.4 DOMAIN FILTERING

Initial experiments using Tranco show that there are many poor quality websites in the list. Some are known to spread malware, host porn, are unreachable or serve no content at all. Some are in a foreign languages other than English or Dutch and are incomprehensible without the use of a translation service. When using a fully automated solution such as a crawler this may be acceptable, but when manually inspecting websites it is not. The main goal of pre-filtering domains is to ensure that the time and effort of the reviewers is utilized as efficiently as possible, and to minimize exposure to malicious content. Therefore, domains are progressively filtered using the conditions below. A domain is discarded from the bucket as soon as any of the checks fails.

1. Domain validation
The domain is parsed and checked for validity by the tldts[10] package. A valid domain is defined as a string containing only an alphanumerical sequence of characters, dots, and dashes. It may not start or end with a dot or dash, and no consecutive dots or dashes are allowed. The domain must end with a valid suffix (such as .nl or .com).

2. Public DNS resolvers
Websites may host inappropriate or unsafe content, for which domains are checked using 9 different secure DNS resolvers (NextDNS, Quad9, OpenDNS Family Shield, Cloud-

[10] `https://www.npmjs.com/package/tldts`

flare Malware+Adult, CleanBrowsing, Comodo Secure DNS, Neustar Family Secure, AdGuard Family Protection, and ControlD Family Friendly). The reason for selecting multiple DNS resolvers is that an initial evaluation of a list of known malicious domains revealed that their results vary considerably. The domain is discarded from the bucket if any of these resolvers returns a negative recommendation.

3. IP address data
Some websites are hosted on servers with IP addresses that are known to be a security risk. When the domain name resolves, the resulting IP addresses are checked using Open Source Intelligence (OSINT) feeds (Squidblacklist, Bluetack, Malc0de, DShield, Spamhaus, Yoyo, Emerging Threats, CINSScore, and Mirai). If any of the IP addresses belonging to the domain is in any of the feeds, the domain is considered a security risk and discarded from the bucket.

4. Reachability
Many websites turn out to be unreachable or redirect to other domains. When a website does not load within five seconds or redirects to an unrelated domain, it is considered unreachable and discarded from the bucket. An unrelated domain is defined as a domain of which the original domain is not a substring. This implies that redirects to other subdomains such as *www* are allowed.

5. Content
There are websites that only contain placeholders, 'coming soon' pages, or are completely empty. After manually evaluating a sample set from Tranco, it was found that webpages roughly containing at least 200 visible words can be considered non-empty. The average word length in English is approximately 4.7 characters[11], thus non-empty pages should have at least $200 \cdot 4.7 \approx 1,000$ characters of visible text. Moreover, to be able to interact with cookie dialogs, the content must be in a language that reviewers understand. Although language detection is possible in short fragments, it is more reliable in longer texts. If a website's readable content length is lower than 1,000 characters or written in a language other than English or Dutch, the domain is discarded from the bucket.

6. Iframes
One of the metrics of the experiment is the number of clicks required in the interaction with a cookie dialog to give or deny consent. Browser plugins can inject JavaScript into a page to detect clicks (using event listeners). Some cookie dialogs use iframes, but event listeners cannot be added to content loaded inside an iframe. This is a result of the same-origin policy enforced by browsers to restrict how scripts can interact with resources on other websites.[12,13] Therefore, the source code of a website is parsed, and if it contains the string 'iframe' the domain is discarded from the bucket. Excluding all websites that contain iframes is an extreme measure that may have implications for the randomness of the domain selection and thus the generalizability of the results, which is further discussed in Section 4.3.5.

Appendix B lists the final selection of domains after applying filters and creating buckets.

[11] `https://norvig.com/mayzner.html`
[12] `https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy`
[13] `https://stackoverflow.com/a/25098153`

4.3.5 LIMITATIONS

There are certain limitations that may affect the outcomes of the experiment. It is important to be aware of these limitations when interpreting the results.

- The selection of domains is pre-filtered. For most of the filters this does not negatively affect the randomness. However, if a certain category of websites solely relies on iframes for their consent dialogs, it is possible that these websites are systematically excluded from the buckets. This poses a threat to the validity of the outcome, as it may negatively affect the generalizability of the results. Observing the effects in a manual evaluation revealed that approximately 30% of the domains is discarded by the iframe-filter, but no specific patterns or categories could be identified.
- When evaluating websites, reviewers indicate the status of the visit by manually selecting a 'reason' (further specified in Section 4.4.2). There is a risk that reviewers evaluate situations differently, which may lead to inconsistent results. To reduce this risk, the reviewers have jointly evaluated a set of websites to align their evaluations as much as possible.
- Websites regularly change, which may also impact the placement of cookies and thus the outcomes of the experiment. Furthermore, cookie dialogs are subject to change due to evolving regulations. This makes it difficult to get reproducible results, and it should be taken into account that the results are a representation of the situation at one point in time.
- Content on websites is not always controlled by website owners. Advertisement space can be sold to various external parties and the placement of cookies can vary between visits. Advertisements may also be dynamically interspersed at a certain time interval during a single visit, resulting in an increasing number of cookies while a webpage is open. Similar complications occur when websites employ A/B testing that also involves the placement of cookies. These circumstances limit the possibility to get reproducible results.
- The results of DENY BASIC and DENY ADVANCED may not be directly comparable, because some websites only offer one of these actions whereas others offer both.
- The external data sources used to determine the purpose of a cookie may produce inconclusive results. One source may categorize a cookie differently than the others, and for completeness it has been decided that both purposes count. When a cookie's purpose cannot be determined, it is labelled as *Unknown* and not counted towards the number of violations. Section 4.7 discusses alternative approaches to handling unclassified cookies.
- Another limitation, or risk, concerns the manual cookie dialog interaction in the DENY ADVANCED mode. Some cookie dialogs are so complicated that it is not always clear whether all possible options have been deselected. When there are multiple possible paths, there may be a difference in click count between reviewers due to the manual nature of the cookie dialog interactions. To reduce this risk, a total of 10 recordings is checked in this mode by a second reviewer.
- Some websites make use of CMPs to collect and propagate cookie preferences to third parties. When a website is reachable but their CMP is down, there is a risk

that the cookie preferences are not saved correctly. This risk is partly mitigated by conducting the experiment over a period of several weeks and visiting each website multiple times.

- Lastly, not all websites need to comply with European privacy laws. For example, the GDPR applies to "*a company or entity which processes personal data as part of the activities of one of its branches established in the EU, regardless of where the data is processed; or a company established outside the EU and is offering goods/services (paid or for free) or is monitoring the behaviour of individuals in the EU.*"[14] When a website places cookies but does not fall in either of these categories, it does not necessarily mean a violation of the law.

In conclusion, although the data resulting from the experiment is not guaranteed to be without errors, it is considered largely representative of what's on the internet.

4.4 ARTIFACTS

Four supportive tools (artifacts) have been developed to assist with conducting the experiment. The stakeholders are the participants in this project, and the tools are primarily intended to be used within the context of this experiment. However, external parties may be interested in running their own instance of the experiment, and therefore the source code has been made publicly available on Github.[15] The tools have been tested manually to validate that everything works as expected.

1. Domaintool: loads domains and corresponding rankings from a Tranco CSV file, creates buckets according to the bucket composition scheme, sanitizes the buckets in line with the domain quality filtering rules, outputs the results to the terminal, and saves the selected domains to a persistent storage location. This application has no dependencies on the other tools and can be used stand-alone.
2. API: provides an interface for clients such as the browser plugin to communicate with the database. There are endpoints to get statistics, get the next website to visit, report (POST) cookies and clicks, get a list of cookies for which no purpose has been determined yet, and report (POST) cookie purposes.
3. Browser plugin ('Cookie Helper'): an extension for the Chrome browser that facilitates visiting a website in a selected mode, counting the number of clicks while performing cookie dialog interactions, and recording which cookies have been placed during a visit. The plugin is a consumer of the API.
4. Cookiepurpose: fetches cookie names from the API for which no purpose has been determined yet, consults the three external data sources (CookiePedia, CookieDatabase, Open Cookie Database), and POSTs the results to the API.

The API runs on a central webserver, and the domaintool, browser plugin and cookiepurpose applications run locally on the user's machine. The following activity diagram shows

[14] https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/application-regulation/who-does-data-protection-law-apply_en

[15] github.com/koenberkhout/dark-patterns-project

which sequence of actions should be performed to conduct the experiment using the supportive tools, and how the resulting data is exchanged. The experiment starts with the domaintool and finishes as soon as there are no unreviewed domains left in the database.

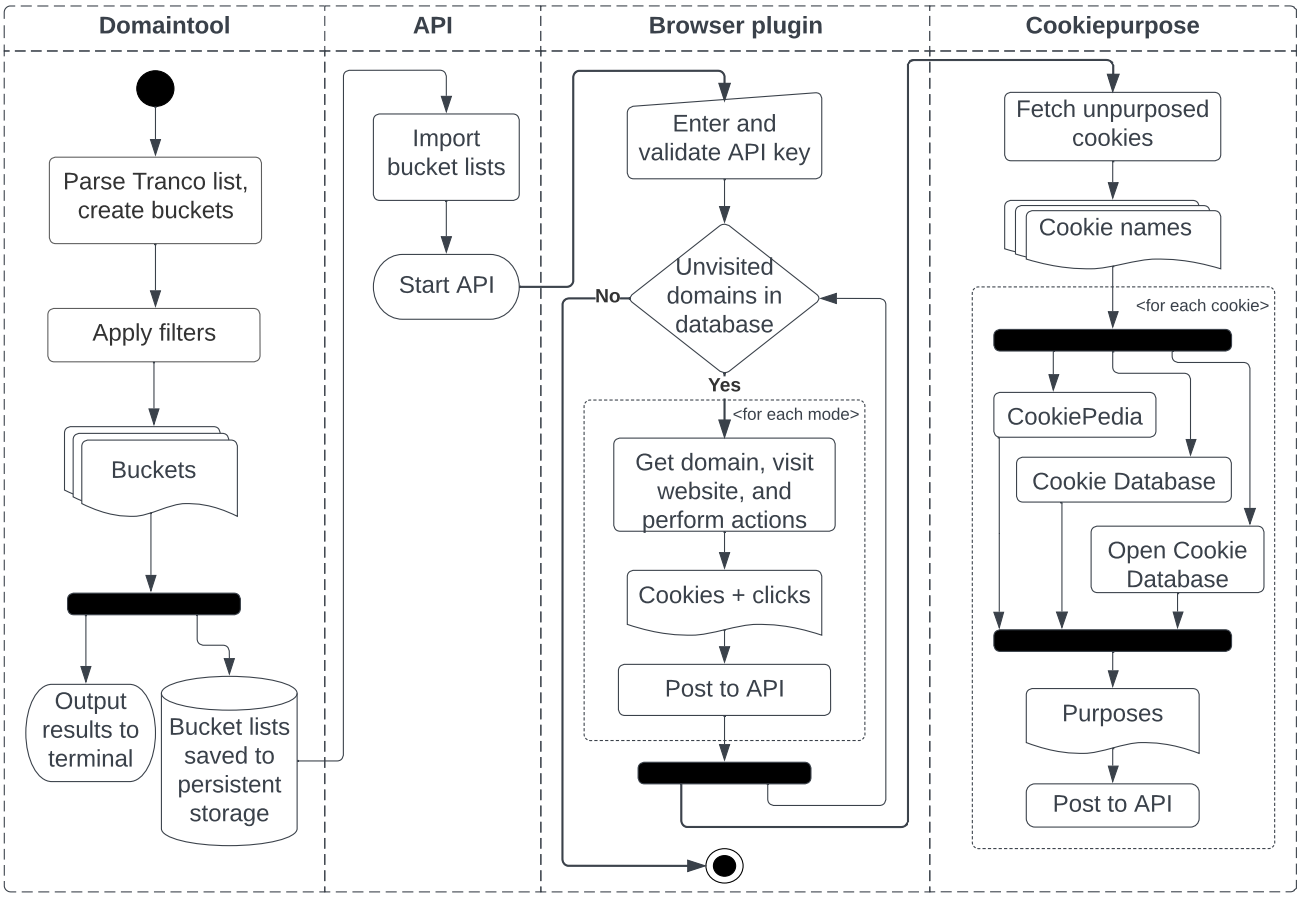


Figure 2: Activity diagram supportive tools

4.4.1 DOMAINTOOL

The domaintool was initially developed as a Bash script, and then rewritten to a Node application using TypeScript which facilitates type checking during development. It is composed of a set of modules (classes) with a manageable scope. When functionality is needed beyond the native Node capabilities, open source libraries are first compared using a set of criteria including age, number of dependencies, development activity, weekly downloads, stars and open issues on Github, and documentation before being selected.

The major challenge in developing this application is *limited* concurrency, because checks need to be performed on potentially a large number of domains while external services such as DNS resolvers may not be flooded with requests. Therefore, domains are checked in chunks of 25 using a rate-limited job queue with the same number of active workers, each of which responsible for processing one job (checking a domain). Jobs are added to the job queue with an interval of 0.5 seconds and get immediately picked up by an available worker. Each job has a time limit of 10 seconds, which is chosen after observing

the average time to complete the evaluation of a domain. The next chunk of domains is processed once all jobs in the current job queue have been completed.

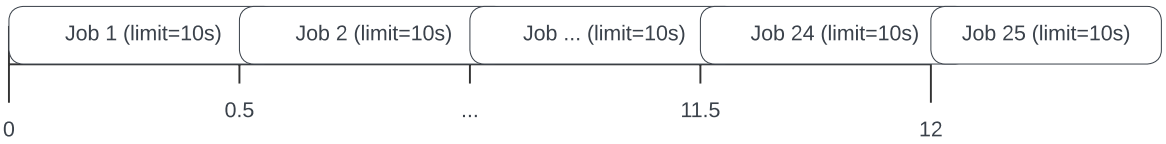


Figure 3: Domaintool job queue timeline

Sequentially checking 25 domains would take at most $25 * 10s = 250$ seconds, whereas with the limited concurrency approach it takes at most $12s + 10s = 22$ seconds. The general formula to calculate the maximum time needed to process a chunk containing D domains with a rate limit of R and a time limit of L seconds per job is $(D * R - R + L) => \mathcal{O}(D * R)$. Other operations such as starting the application are negligible.

The simplified class diagram below shows how the domaintool is composed of different classes. Each class has its own responsibility and is exported as a JavaScript module, which is then dynamically imported where necessary. Some classes only contain static functions ('class methods') and need not be instantiated explicitly, but this distinction is disregarded on purpose because in JavaScript objects are always implicitly created.

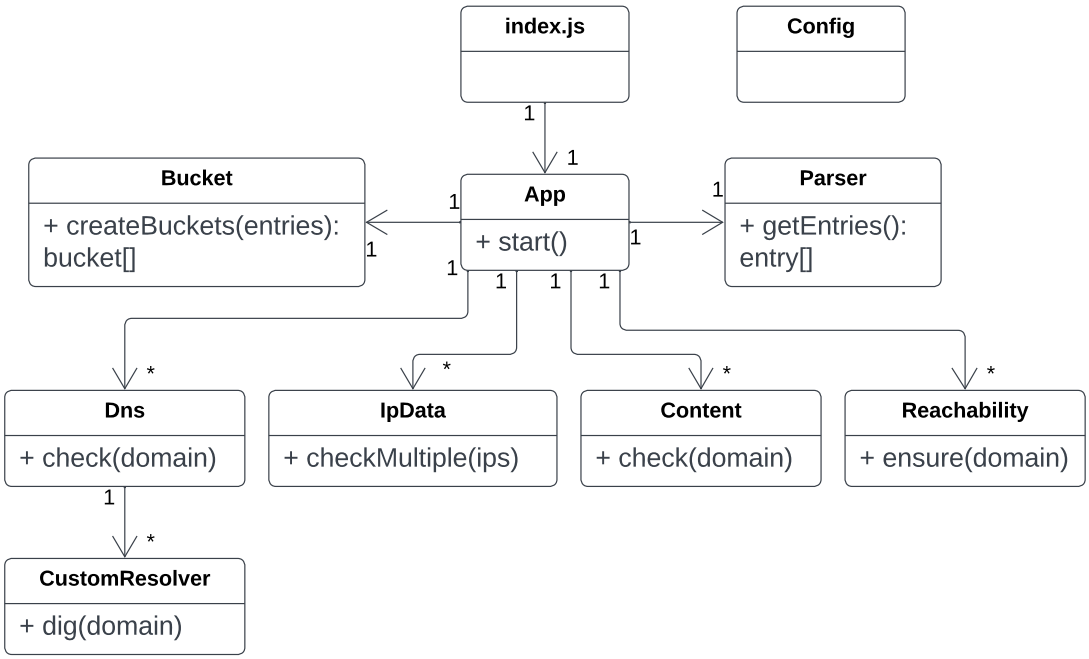


Figure 4: Domaintool class diagram

The collaboration between objects in the domaintool is shown in the sequence diagram in Figure 5. Promises are used where no value is explicitly returned after a function invocation. If any of the checks fails, the promise is rejected and the domain is discarded from the bucket. As with the class diagram, helper functions are intentionally left out.

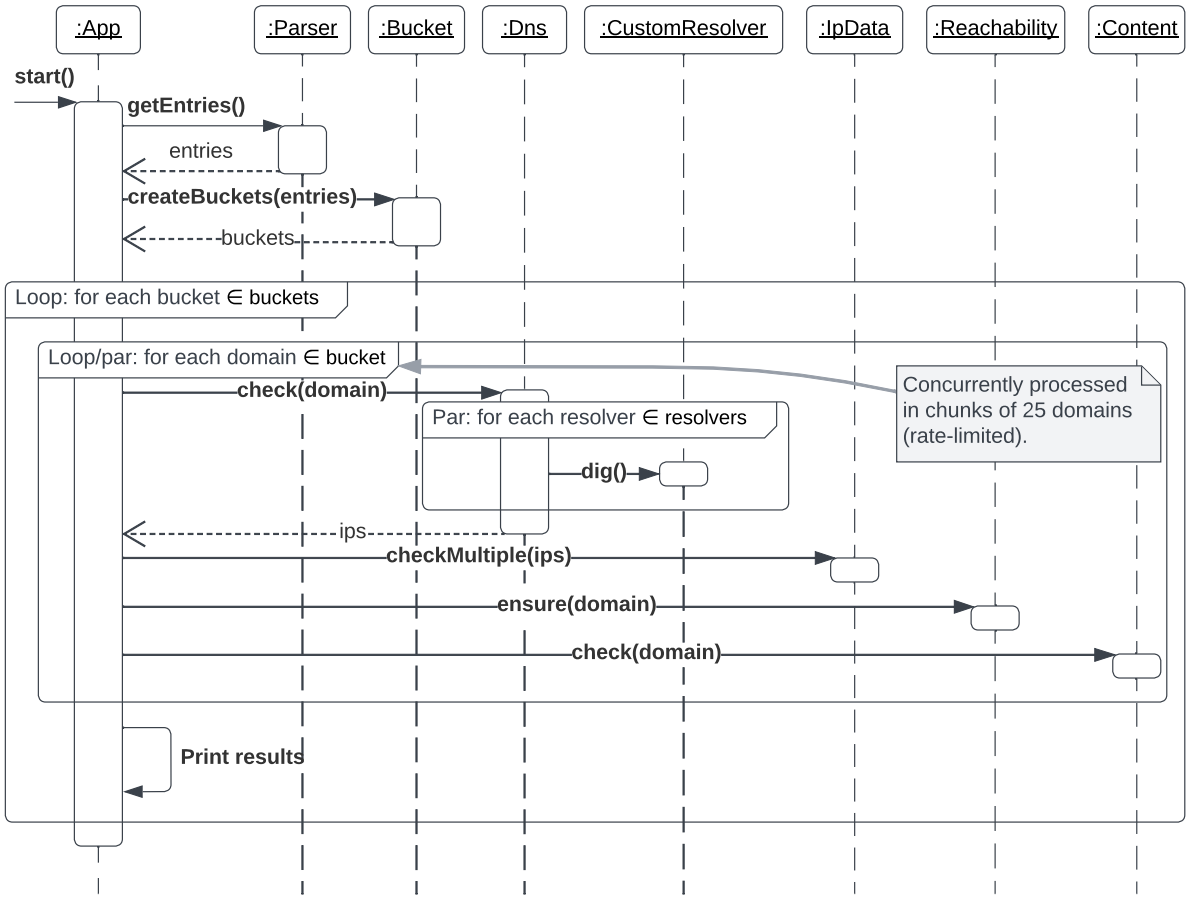


Figure 5: Domaintool sequence diagram

Figure 6 shows the partial results produced by the domaintool after processing one chunk. Domains printed in green are selected for inclusion. Domains printed in red did not pass the filters and are discarded from the bucket.

#	rank		domain	elapsed	0	1	2	3	4	5	6	7	8	ipdata	reachable	parsable	iframeless	content-length	language
1	4982		hwccpc.com	4095 ms	x	x	x	x	x	x	x	x	x	–	–	–	–	–	–
2	414		redd.it	1338 ms	v	v	v	v	v	v	v	v	v	v	x	–	–	–	–
3	976		service-now.com	666 ms	v	v	v	v	v	v	v	v	v	v	x	–	–	–	–
4	3796		viki.com	917 ms	v	v	v	v	v	v	v	v	v	v	v	v	x	–	–
5	3861	🏆	realsimple.com	3424 ms	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v
6	206		rackspace.net	3193 ms	v	v	v	v	v	v	v	v	v	v	x	–	–	–	–
7	1655		creativecdn.com	10 ms	x	v	v	v	v	v	v	x	x	–	–	–	–	–	–
8	3531		magento.com	3497 ms	v	v	v	v	v	v	v	v	v	v	x	–	–	–	–
9	4155		eegeeglou.com	40 ms	x	v	v	v	v	v	v	x	x	–	–	–	–	–	–
10	1096		chicagotribune.com	8680 ms	v	v	v	v	v	x	v	v	v	–	–	–	–	–	–
11	155	🏆	sourceforge.net	928 ms	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v
12	3527		bendibao.com	5060 ms	v	v	v	v	v	v	v	v	v	x	–	–	–	–	–
13	4021		byteoversea.com	139 ms	x	x	x	x	x	x	x	x	x	–	–	–	–	–	–
14	539		sagepub.com	779 ms	v	v	v	v	v	v	v	v	v	v	v	v	x	–	–
15	2426		cvs.com	553 ms	v	v	v	v	v	v	v	v	v	v	v	v	v	x	–
16	1129		cricbuzz.com	116 ms	v	v	v	v	v	v	v	v	v	x	–	–	–	–	–
17	2981		rfi.fr	406 ms	v	v	v	v	v	v	v	v	v	v	v	v	x	–	–
18	3427	🏆	vuejs.org	254 ms	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v
19	935		lowes.com	320 ms	v	v	v	v	v	v	v	v	v	v	x	–	–	–	–
20	532	🏆	trustpilot.com	1733 ms	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v
21	592		hugedomains.com	46 ms	x	v	v	v	v	v	v	v	v	–	–	–	–	–	–
22	3894		psychcentral.com	802 ms	v	v	v	v	v	v	v	v	v	v	v	v	x	–	–
23	3823	🏆	hardened-php.net	1024 ms	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v
24	4031		cnmsn.net	7915 ms	v	v	v	v	v	v	v	v	v	x	–	–	–	–	–
25	1572		drupal.org	547 ms	v	v	v	v	v	v	v	v	v	v	v	v	x	–	–

Figure 6: Domaintool results

4.4.2 Browser Plugin

The Cookie Helper (Figure 7) is a custom-built plugin for Chrome written in JavaScript. Its main purpose is to automate the most time-consuming actions that are involved with reviewing websites and their cookie placement behavior. There are various states the plugin can be in, and actions in the GUI may cause the state to transition to another state. Once the user has entered a valid API key, it moves to the superstate *mode_initial*. In this state, the user visits the website without performing any actions and records which cookies have been placed. In the other modes, the user interacts with cookie dialogs. Both cookies and the number of clicks are recorded. This core functionality has been extensively tested by three reviewers and shown to work correctly. The state machine diagram in Figure 8 depicts the possible states and the actions that can initiate state transitions.

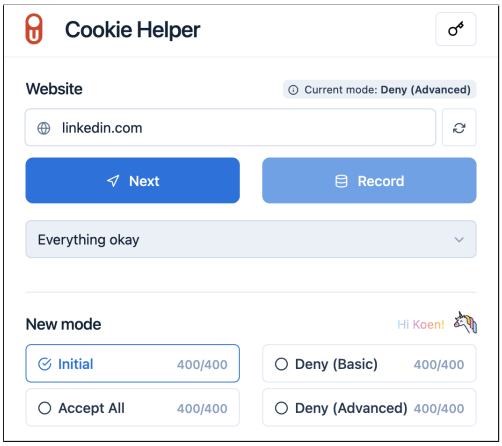


Figure 7: Browser plugin GUI

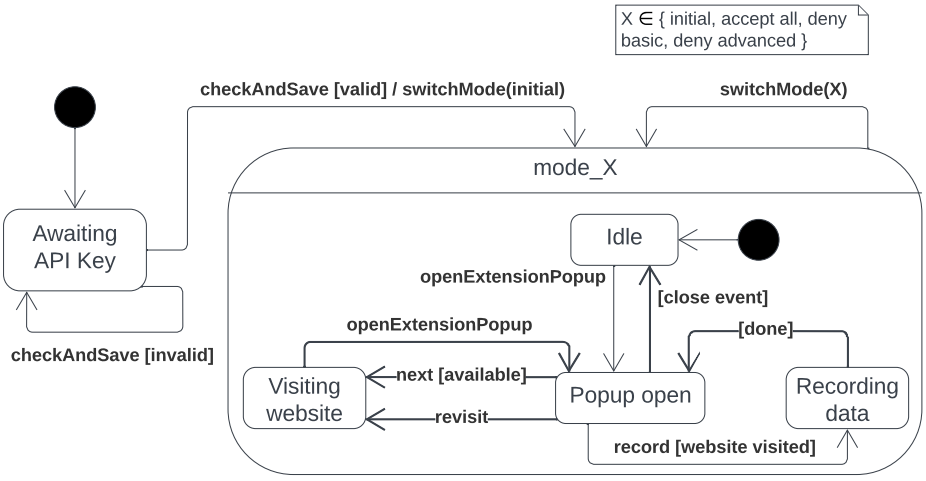


Figure 8: Browser plugin state machine diagram

After visiting a website, but before recording the cookies and clicks, the user must select one of the following *reason*s. This indicates whether special circumstances occurred during a visit.

1. Everything okay (default)
2. Dialog, but action unavailable
3. Notice only
4. No dialog/notice at all
5. Website not loading
6. Landing page (e.g. country selector)
7. Incomprehensible language

It is unnecessary to revisit a website in any of the remaining modes when the selected reason is 3, 4, 5, 6, or 7, because the outcome of the visit is expected to be identical. In these special cases, the API automatically records the result for this website for all of the modes.

4.4.3 API

The API enables multiple remote clients to fetch and store information, and is composed of a MySQL database and a PHP application built with the Fat-Free[16] micro-framework. This framework provides functionality such as a routing engine and database connectivity that is by default safe from SQL injection.

Figure 9 is a model of the database that is used in the API. It is normalized to the extent possible. The *website* table is filled with the domains that are selected by the domaintool. The *reason* table contains the seven reasons introduced in the previous section. When a recording is received from the browser plugin, an entry is inserted into the *recording* table, and then all corresponding cookies are inserted into *cookie* and linked to this recording. The *purpose* table contains all cookie names and purposes insofar as they can be determined by CookiePedia, CookieDatabase and Open Cookie Database.

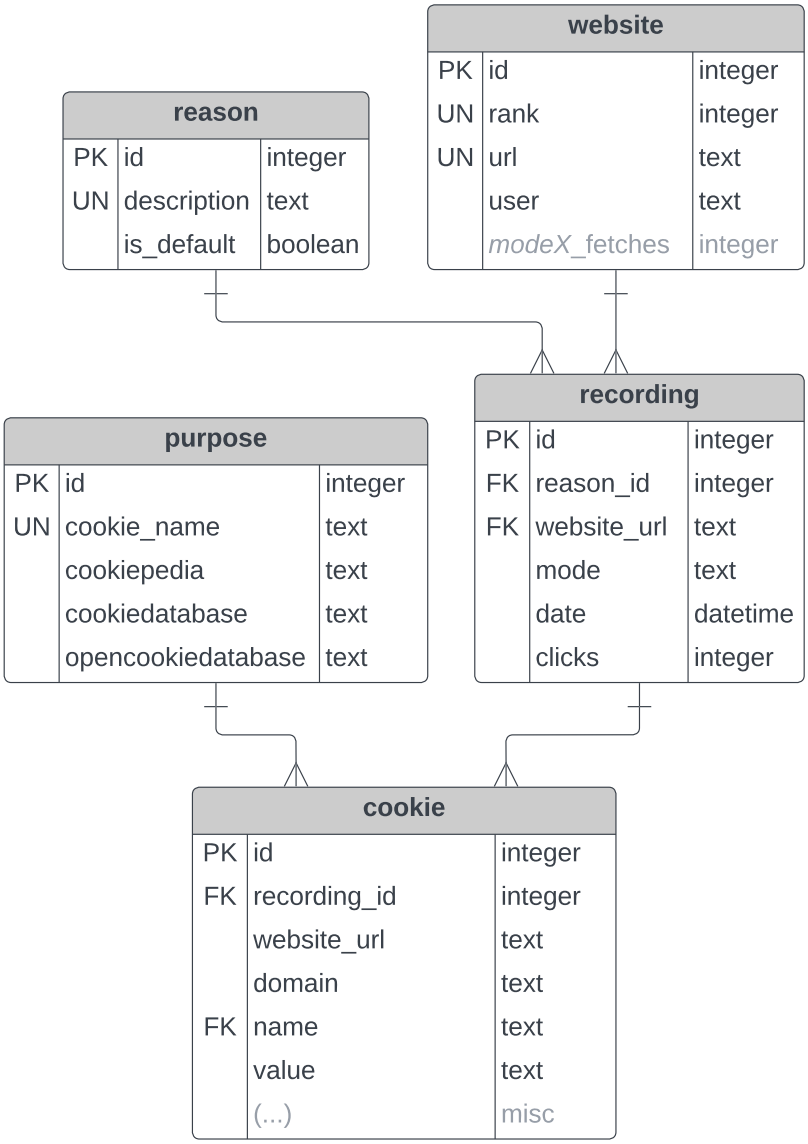


Figure 9: API database model

[16] https://github.com/bcosca/fatfree

There are various endpoints that enable clients to get information from- and add data to the database, which all require authentication. Hence, each call must be accompanied with an API key or the response is 401 Unauthorized. Sensitive information such as database credentials are loaded from a .env file. Endpoints have a JavaScript Object Notation (JSON) response type. The endpoints listed below are defined in the Router class and implemented in the Controller.

- GET /
- GET /stats-reasons
- GET /validate-key
- GET /next-website/@mode
- POST /report-cookies/@mode/@url
- GET /unpurposed-cookie-names
- POST /report-cookie-purposes/@which

4.4.4 COOKIEPURPOSE

Once cookies have been added to the database as part of a recording, their names can be fetched from the */unpurposed-cookie-names* endpoint. Similar to the domaintool, the cookiepurpose tool is a Node application written in TypeScript. It uses the cookies' names to determine their purpose by concurrently consulting CookiePedia, CookieDatabase and Open Cookie Database. Getting purpose data requires a different approach for each of the external sources, which is reflected in the design of the tool. Whereas reading entries from the CSV file provided by the Open Cookie Database is straightforward, getting data from the other sources requires more effort.

CookiePedia
Although CookiePedia has the largest cookie database, they offer no possibility to programmatically access the data. An email was sent to request access to an API (if any) or downloadable file, but the message remained unanswered. Therefore, CookiePedia is scraped using the Chromium automation tool Puppeteer[17]. It is equipped with the Stealth[18] plugin to avoid detection, as some websites employ bot detection techniques to block crawlers. The setup has been validated manually to ensure that the scraper works as expected. This is indeed the case, either because CookiePedia does not perform bot detection or because the scraper could not be detected.

Cookie Database
Similar to CookiePedia, Cookie Database does not have a publicly available API. A possible workaround would be to scrape the website, but the Complianz[19] WordPress plugin was found to use an undocumented API on cookiedatabase.org that allows for determining cookie purposes in bulk. Complianz is one of the maintainers of the Cookie Database project. A solution was found after reverse-engineering the open source plugin to discover the required request format, which is reflected in the *CookieDatabase.composeRequestOptions()*[20] function.

[17] `https://github.com/puppeteer/puppeteer`
[18] `https://github.com/berstend/puppeteer-extra`
[19] `https://nl.wordpress.org/plugins/complianz-gdpr/`
[20] `https://github.com/koenberkhout/dark-patterns-project/blob/main/cookiepurpose/src/CookieDatabase.ts#L37`

The class diagram in Figure 10 models how the cookiepurpose application is composed.

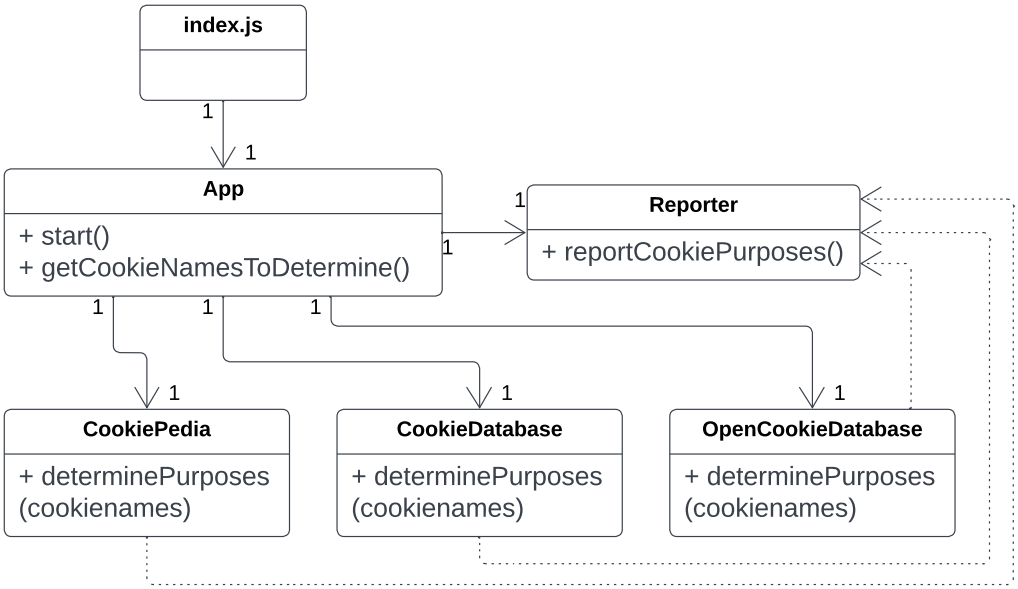


Figure 10: Cookiepurpose class diagram

4.5 Results

After visiting the 400 selected websites, it was found that the majority does not show any cookie dialog or notice. The complete distribution of found reasons is given in Figure 11. Note that all reasons are mutually exclusive except 1 and 2, thus the domain count is $|1 \cup 2| + |3| + |4| + |5| + |6| + |7| = 400$. Reasons 1–4 indicate a successful visit, and reasons 5–7 indicate an unsuccessful visit. If the reason is 3 or 4, the results are the same for each visiting mode and only the INITIAL mode is considered.

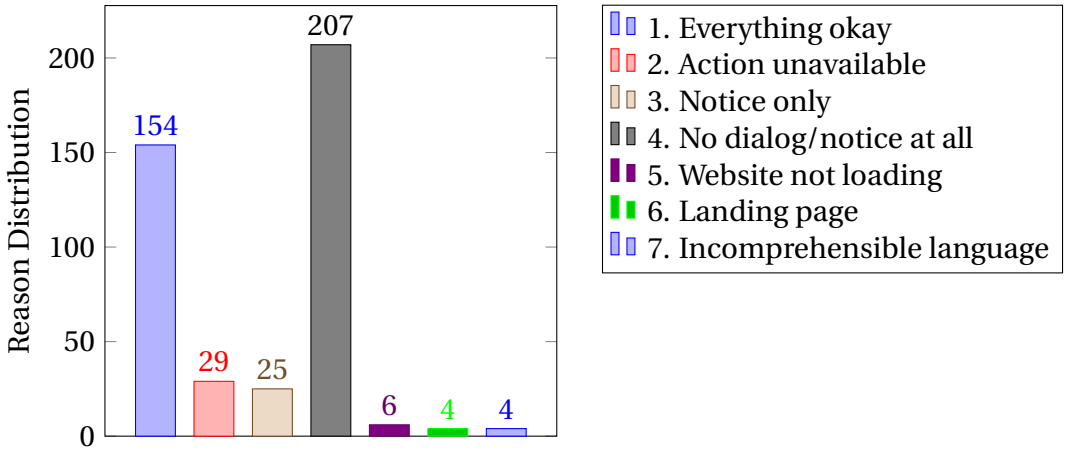


Figure 11: Reason distribution

Henceforth, all domains with reasons 5–7 will be ignored. The valid result set for the initial mode consists of the 386 remaining domains. Successful interaction with a cookie dialog is required in the other three modes, thus only domains with reason 1 are included. The result count is the total number of domains with a valid result for a specific mode. None of the visited websites with a DENY BASIC action offered DENY ADVANCED and vice versa. Table 4 lists the result count for each mode.

Mode	Result count
INITIAL	386
ACCEPT ALL	154
DENY BASIC	126
DENY ADVANCED	134

Table 4: Result counts

4.5.1 Cookie Counts

Figure 12 shows the average number of cookies placed by the visited websites, rounded to the nearest integer. It is apparent that ACCEPT ALL results in the greatest number of cookies, but the other modes do not differ much from each other. This indicates that cookie dialogs are likely to be effective when the user wants to give consent, but also that denying consent does not result in a lower number of cookies than initially placed.

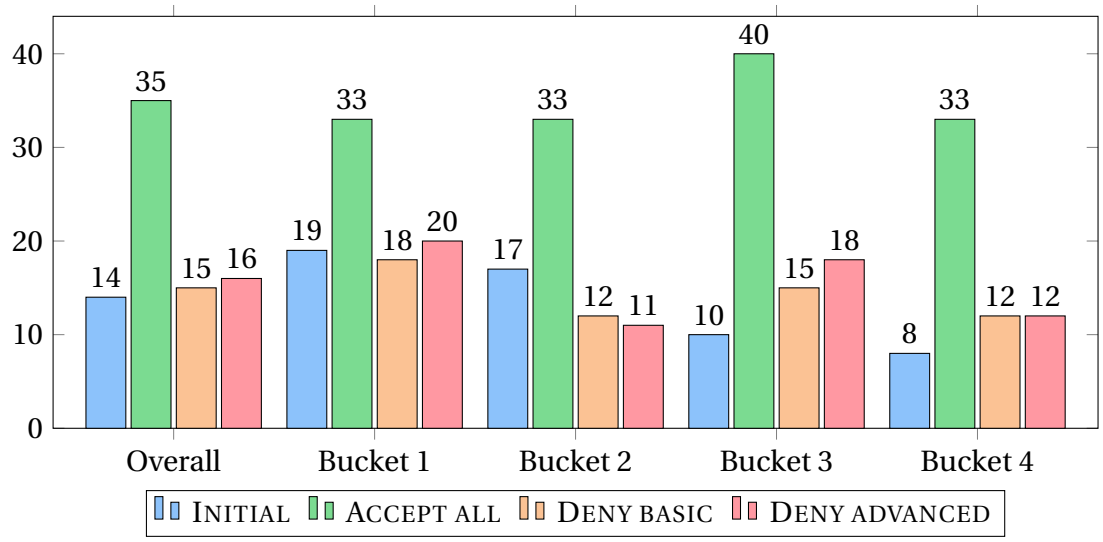


Figure 12: Average cookie counts

There are outliers in the result set that cause the average cookie counts to give a distorted impression of the number of cookies placed in the various modes. The cookie count distribution in Figure 13 shows these outliers and compares the results of each mode. The left border of the box is the first quartile (Q1) and the right border is the third quartile (Q3). Values higher than $Q3 + 1.5 \cdot (Q3 - Q1)$ are considered outliers. Similar to the average cookie counts, the distribution suggests that giving consent is probably effective, but rejecting cookies in any of the DENY modes does not result in a substantially lower number of cookies than initially placed.

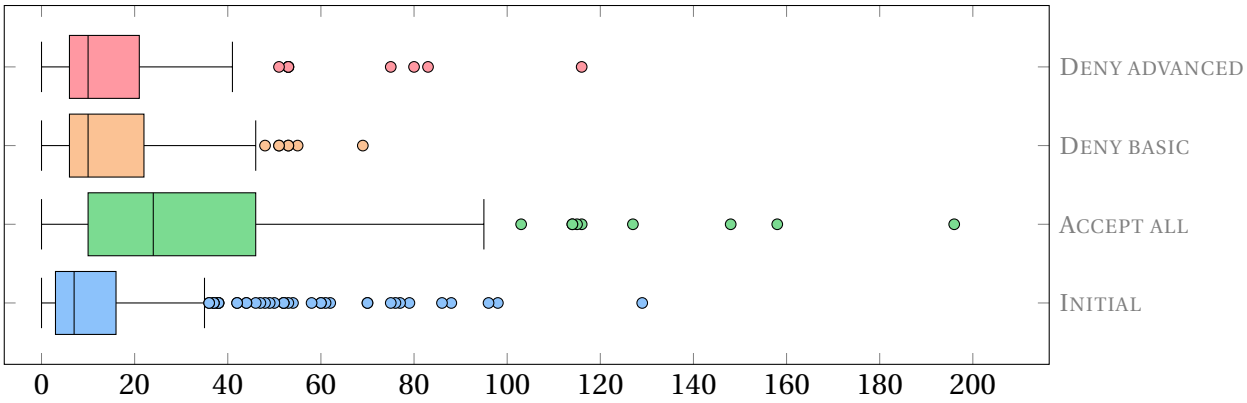


Figure 13: Cookie count distribution

4.5.2 COOKIE PURPOSES

Although the cookie count is the largest after ACCEPT ALL, there is still a significant number of cookies placed in the other modes. To determine the purpose of these cookies, the categorizations of the three external sources are consulted and mapped according to Table 1. If the purpose can be determined, it is classified as either Necessary, Performance, Functionality, or Marketing. As discussed in Section 4.2, only necessary cookies are exempted from explicit user consent. Table 5 lists the average number of cookies for

each mode by purpose and bucket. It seems that the number of cookies decreases when progressing from bucket 1–4. As can be seen in Figure 12, this is not necessarily true.

	#Necessary	#Performance	#Functionality	#Marketing	#Unknown
INITIAL					
Overall	2.4	3.9	0.9	4.1	2.9
Bucket 1	3.2	4.3	1.5	5.0	5.7
Bucket 2	2.7	4.7	1.4	5.7	2.9
Bucket 3	1.7	3.2	0.5	3.2	1.7
Bucket 4	1.7	3.4	0.3	2.5	1.0
ACCEPT ALL					
Overall	4.9	6.4	1.7	14.3	8.1
Bucket 1	5.7	6.6	2.3	11.7	7.7
Bucket 2	4.7	6.7	1.4	14.1	6.9
Bucket 3	4.5	5.9	1.6	18.1	10.1
Bucket 4	3.5	5.8	0.8	16.1	8.2
DENY BASIC					
Overall	3.4	3.4	0.8	3.1	4.9
Bucket 1	3.8	3.2	1.2	3.6	6.7
Bucket 2	3.3	3.3	0.5	1.9	3.4
Bucket 3	3.4	3.3	0.7	3.6	4.3
Bucket 4	2.3	3.4	0.1	3.5	2.5
DENY ADV.					
Overall	3.6	3.7	0.9	3.7	4.9
Bucket 1	4.2	4.2	1.3	4.0	7.1
Bucket 2	3.2	3.2	0.6	1.8	2.8
Bucket 3	3.4	3.2	0.9	5.5	5
Bucket 4	2.4	4.1	0.1	3.4	2.4

Table 5: Cookie purposes by type

From this table it appears that denying consent results in a small increase in necessary cookies and a small decrease in unnecessary cookies (performance, functionality, marketing). In the INITIAL mode, only 52 out of 386 websites do not place any unnecessary cookies, which is a failure rate of approximately 86.6%. Figure 14 shows the distributions of the unnecessary cookies that are placed in the various modes.

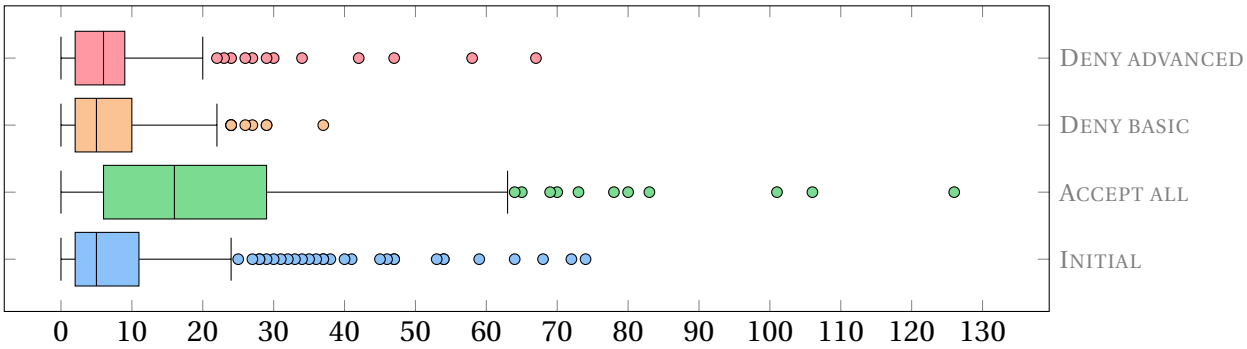


Figure 14: Unnecessary cookies distributions

4.5.3 CLICK COUNTS

Having experienced the proliferation of cookie dialogs after the GDPR entered into force, the first intuition is that it is considerably easier to accept all cookies than to reject them. To test this assumption, the Cookie Helper tool recorded the number of clicks required to interact with a cookie dialog in addition to the number of cookies placed. A visit is successful when the action belonging to a mode can be successfully performed. Although four distinct actions have been defined, doing nothing in INITIAL is an empty action for which the click count is always zero. Thus, the following comparison only considers accepting and denying consent. Table 6 lists the average number of clicks needed to perform the action that corresponds with a certain mode.

Mode	Overall	Bucket 1	Bucket 2	Bucket 3	Bucket 4
ACCEPT ALL	1.1	1.2	1.0	1.1	1.3
DENY BASIC	2.1	1.7	2.1	1.9	3.8
DENY ADVANCED	33	16.9	56.6	45.7	6.1

Table 6: Average click counts

Whereas ACCEPT ALL and DENY BASIC do not require many clicks, DENY ADVANCED takes more effort to complete. There is a large difference between the buckets in this mode, and analyzing the individual click counts reveals that a few extreme outliers strongly influence the averages. Table 7 lists the average click counts without outliers for each bucket.

	Overall	Bucket 1	Bucket 2	Bucket 3	Bucket 4
Average	6.8	7.1	8.3	6.6	6.1
Outliers	any > 31	254,73,67,55,51,39,38	1265,286,262	876,151,126	none

Table 7: Average click counts without outliers for DENY ADVANCED

Figure 15 shows the overall click count distribution and the distributions partitioned by bucket for DENY ADVANCED. The outliers have been omitted for readability.

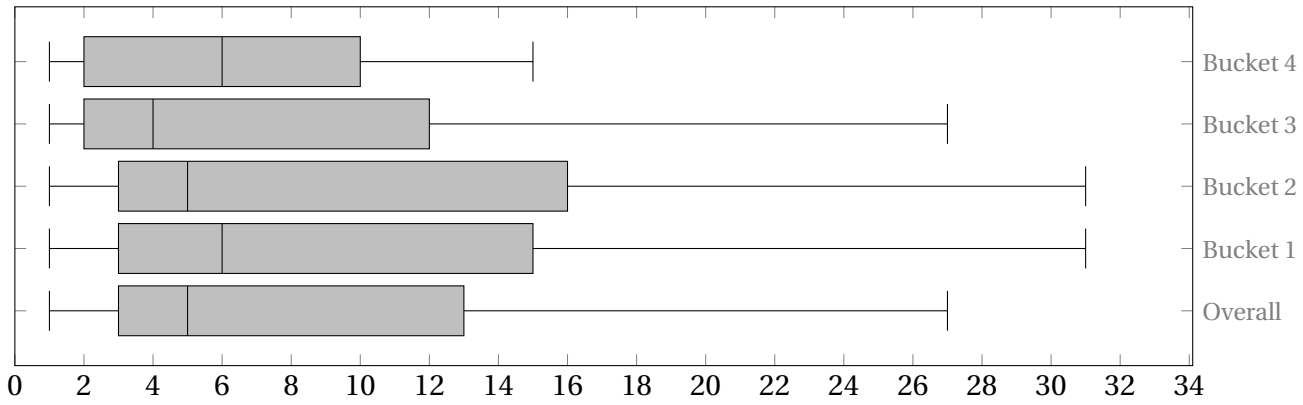


Figure 15: Click count distributions without outliers for DENY ADVANCED

Counting the required number of clicks in this mode was identified as a risk in Section 4.3.5 due to potential differences between reviewers. Therefore, the number of clicks submitted by reviewer 2 and 3 are each validated by reviewer 1 for five visited websites. These websites are randomly drawn from the result set.

Set A: { mysql.com=10, bbc.com=67, venturebeat.com=39, discogs.com=51, cc.com=14 }
Set B: { reuters.com=15, espn.com=55, livestrong.com=73, telenet.be=27, sixt.com=12 }

Set	Reviewer	AVG	AVG(Δ)	AVG(%Δ)	MAX(Δ)
A	2	36.2	0.7	1.9%	3
B	3	36.4	1.1	3.0%	5

Table 8: Click count validation

The average number of clicks is 36.3 and the average absolute difference in clicks is 1.1. The maximum absolute difference found is 5 (livestrong.com). The deviation is an acceptable 2.5% on average.

4.6 DISCUSSION

The results reveal that in general there is a marginally higher cookie count after rejecting cookies than initially placed. This does not necessarily mean that doing nothing is more effective than rejecting cookies. It was found that in most cases websites save the user's cookie preferences by placing one more 'preference' cookie. This indicates that doing nothing is as effective as rejecting cookies. In some cases, however, the purpose of the newly placed cookie could not be determined nor inferred.

Although doing nothing is similarly effective as rejecting all cookies, this does not mean that there are no violations. On average, 8.9 unnecessary cookies are placed before any cookie dialog interaction has taken place. This is slightly reduced to 7.3 and 8.4 in DENY BASIC and DENY ADVANCED, respectively. These numbers are probably on the low end, because they do not include cookies whose purpose could not be determined. In an ideal situation, the number of unnecessary cookies should be zero.

The observed failure rate is approximately 86.6%, while the study by Bollinger et al. found a failure rate of almost 95% of the websites. Ideally, the situation has improved with nearly ten percent, but Bollinger focused exclusively on websites with a CMP which may cause an unfair comparison. Whereas Sanchez et al. refer to the placement of tracking cookies despite denying consent as 'false rejections', it is now found that these cookies were probably already placed on page load. They also observed that the majority of the web pages place unnecessary cookies even if users do not give consent, which is confirmed by the findings in this substudy. Furthermore, they found that rejecting tracking is usually ineffective, which is substantiated by the non-decreasing cookie counts in DENY BASIC and DENY ADVANCED in Figure 12 and Figure 13. What appears to have improved in the last three years is the ease at which users can (seemingly) opt out from tracking. Sanchez found that only 4% of the website have a clear DENY BASIC option, whereas Table 4 indicates that currently nearly 33% of the websites provides such option.

Table 5 suggests that higher-ranked websites place more cookies than websites with lower rankings, but this observation is contradicted by the total cookie counts in Figure 12. It is clear that websites with a high ranking place more cookies in INITIAL than websites with a low ranking, but the other modes do not indicate a clear trend. Note that one of the limitations of this study is the use of three external data sources to determine the purpose of a cookie, and that the resulting purpose data may overlap.

There is a strong resemblance in the cookie placement behavior between the modes after rejecting cookies. However, the effort needed to complete the actions varies greatly. It requires on average 2.1 clicks to complete DENY BASIC, whereas DENY ADVANCED requires on average 33 clicks. As mentioned in the results section, none of the visited websites with a DENY BASIC option offer DENY ADVANCED and vice versa. The majority of the visited websites only present the user with a complicated cookie dialog, making it unnecessarily difficult to reject all cookies. There is a strong imbalance between accepting and rejecting cookies with on average 1.1 : 33 clicks, indicating the presence of the *obstruction* dark pattern.

4.7 FUTURE WORK

One of the limitations of this study is the usage of three external data sources for determining cookie purposes, with the advantage that there is more purpose data available. All purposes count when results are inconclusive, which may lead to inconsistent results. It would be useful to analyze the differences between the external sources, and contribute to improve the cookie data by submitting corrections.

As discussed in Section 4.3.5, the selection of domains is pre-filtered. Whereas most of the filters do not greatly affect the randomness, the restriction that all websites containing an iframe be excluded may negatively affect the generalizability of the results. This was a concession made to be able to count the number of clicks while performing cookie dialog interaction. For additional validation of the results, the experiment should be redone without using the restrictive iframe filter.

Cookies for which the purpose could not be determined do not fall in the categories necessary or unnecessary and are thus not taken into consideration. Future research could think of ways to determine their purpose, for example by scraping on a larger scale or by making predictions based on calculations such as the entropy of a cookie value.

The GDPR only applies to "*a company or entity which processes personal data as part of the activities of one of its branches established in the EU, regardless of where the data is processed; or a company established outside the EU and is offering goods/services (paid or for free) or is monitoring the behaviour of individuals in the EU*"[21]. Being able to automatically determine whether a website meets these requirements would improve the generalizability of conclusions drawn based on the types of cookies placed by websites.

The artifacts that are created to support the experiment have undergone thorough manual testing. An important engineering aspect that has been omitted due to time constraints is automated testing. The tools will not be deployed to an external production

[21] https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/application-regulation/who-does-data-protection-law-apply_en

environment. Nonetheless, if the tools prove helpful to other projects and be used on a larger scale, it is recommended to first make a test plan and create automated tests. This simplifies refactoring and the addition of new functionality without introducing breaking changes, and may increase comprehensibility for developers.

4.8 Conclusion

In this substudy, 400 websites have been reviewed using custom-built software tools to determine their compliance with cookie preferences. One of the findings is that doing nothing and rejecting cookies are similarly effective. Therefore, users are recommended to hide cookie dialogs, for example by installing a browser extension that can hide elements in the content of the website. Although websites should not place unnecessary cookies without consent, on average more than 8 of this type of cookies are placed on each visit. It is therefore advisable that DPAs increase their level of surveillance and impose sanctions to protect EU citizens from being excessively tracked. The large number of clicks required to complete DENY ADVANCED suggests that data protection authorities have righteously started imposing fines on companies that do not offer the possibility to reject cookies with the same effort as to accept them.[22,23] Here too, based on the findings in this study, it is recommended that DPAs enforce more stringently and set examples for other companies to do better. Given that each EU member state has its own DPA allows for discrepancies among enforcement between countries. In theory, companies could move from e.g. France to Ireland to avoid fines. Therefore, it is worth considering to integrate all national DPAs into one powerful European DPA.
According to the definition in Section 4.2, websites comply with cookie preferences if they do not initially place unnecessary cookies and adhere to the cookie preferences stated by a visitor, regardless of intentions. It has been found that the vast majority of the websites does not satisfy this criterion. All this considering, the conclusion arises that many websites do not comply with cookie preferences, and there is still a long way to go.

[22] https://www.theverge.com/2022/4/21/23035289/google-reject-all-cookie-button-eu-privacy-data-laws
[23] https://www.cnil.fr/fr/node/122962

5 CRAWLING WEBSITES FOR COOKIES AND COOKIE DIALOGS

5.1 INTRODUCTION

In the substudy in Section 4, 400 websites were manually visited. Although 400 websites gives an insight into the behavior around cookie dialogs, the manual nature of this research limits the scaleability and elevates the risk of user error. In this substudy a larger amount of websites is visited in an automated fashion, withing the limitations of the resources available (be it time, bandwidth or other), to get a broader view of the internet landscape. This will give an unbiased view, eliminating the chance of user error or user bias that could possibly alter the results.

Using the data gathered, an analysis on how many websites do or do not show a cookie dialog and what cookies were set is done. Using a ranked list will give an idea if there are differences between more and less visited websites. The same applies to the options that are showed in a cookie dialog. How many website show which options to the user, and what happens after interacting with an option in a cookie dialog will give an insight into what actually happens behind what the user can see, which and how many cookies are being set before any interaction is done and what cookies are set or altered when a user gives or denies consent.

Information about the design and wording of the cookie dialog, and its options, is also saved and can give an insight into what designs choices are made. Future work will be able to use this information to categorize each website as to what special designs are used to get users into accepting something the website wants instead of what the user wants.

5.2 RESEARCH

To visit a large list of websites in an automated fashion a few questions are to be answered:

How to performantly scale visits
Manually visiting websites is a good option if a small list of websites needs to be visited. Each visit can be assisted by an offloading program that saves relevant information as is done in Section 4. However, visiting each website manually takes time and can only be scaled by adding more users, but each single user can only visit a certain amount of websites and adding more users could add more user error as well. Scaling to a lot more websites in an automated fashion will give a more profound unbiased analysis of the current online landscape.

In this substudy 100,000 websites are visited in an automated fashion. This number was deemed a good balance between visiting enough websites and being executable given the resources available. Especially time and bandwith available were the biggest limiting factors. Given the performance of current hardware and software, this can be done without user interaction by a home computer. Although the answer seems simple, use an automated browser and run multiple session concurrently, the answer is harder than it looks. Keeping resource usage controllable and saving all information orderly and without failure was necessary to avoid crashes that could stop the process prematurely. Work on keeping the automated visit undetectable was also needed to limit websites from showing a different website than what a user could see, set different cookies, or even blocking

access to the website.

How to detect a cookie dialog in an automated fashion?
In this substudy the presence of a cookie dialog will be automatically detected on each visited website. Websites that do or do not show a cookie dialog will be compared to the cookie being set.

Visiting a website in an automated fashion without user interaction is fairly straightforward, as is the detection of cookies being set. Detecting the presence of a cookie dialog is harder, as is already done by other papers (like [Aer21] and [EN16]). A cookie dialog can be implemented in numerous ways and the nature of this makes is hard to detect automatically.

How to detect the options in a cookie dialog?
In this substudy the presence of options available in each detected cookie dialog will also be automatically detected. Eventually the options are to be interacted with, the design information of the options and the cookies set after interaction are to be saved. The detection of the options will show what options each website makes available to the end user and what cookies are being set after interaction.

5.3 Experiment

5.3.1 Concept

1. Purpose of the experiment
The purpose of the experiment is to obtain, from a large list of websites, information around the presence and design of the cookie dialog and the cookies being set. This information from each website will tell if a cookie dialog is used, what options are present on a cookie dialog, what and how many cookies are set with or without a cookie dialog, and what and how many cookies are set before or after giving or denying consent.

2. How will the experiment work
This experiment will work by visiting a large list of websites in an automated fashion, running fast and reliable. During each visit the presence of a cookie dialog and its options is detected. Using the detected options user interaction is executed, simulating a user clicking on an option. Before and after interaction a screenshot, cookies, and design information of the cookie dialog and its options is saved to an SQL database.

3. Automated browser technology
The experiment uses an automated browser to visit a large list of websites. The experiment uses multiple concurrent sessions where each session will spawn a browser instance and visit a website. There are a few different automated browser solutions the could have been used here. Pyppeteer[24] was extensively tested but not a valid options since Pyppeteer can not enter iframes which was used by around 5% of the cookie dialogs. OpenWPM[25], used by Englehardt et al. [EN16] was also extensively tested using coding already used from previous tests. Although it was a very performant solution that out of the box saves a lot of information that was needed, because of the framework around it, it was determined it would be too difficult to implement all the features needed and

[24] https://github.com/pyppeteer/pyppeteer
[25] https://github.com/openwpm/OpenWPM

development was halted.

Selenium was eventually chosen for the final version. Previous coding efforts could be reused here as well. Selenium is perhaps the most used automated browser and a lot of documentation and problem solving can be found online. Selenium-wire[26] was also tested to easily detect redirects and easily block images and other bandwith hungry files. It was used for a full run, but afterwards it was determined there were a lot less cookies being detected and this was therefore repaced by the normal Selenium version. Selenium however, comes with one mayor issue that needs to be addressed during the runs. If a browser session is not shut down correctly, some processes and temporary data stay behind. If too many sessions were not exited correctly the lingering processes will tax the CPU until the system starts halting. The temporary data could fill up the hard disk stopping further sessions. Both of these as well as running to many concurrent sessions at the same time only results in failed visits, without a clear message to the user.

4. Detecting a cookie dialog
Some websites use a Consent Management Platform (CMP) to offload the process of creating and maintaining a cookie dialog. Hardcoding detection information from a CMP to automatically detect a cookie dialog remains insufficient because each and every website that integrates a certain CMP can usually customize it to their liking. CMPs can give multiple options to codewise integrate their cookie dialog into the website and every website can decide to tweak the implementation as well.

Detecting certain keywords comes with similar problems, each website can change the wording in a cookie dialog, the logos that are used, and the cookie dialog can be implemented in the language of the website. Figure 16 shows a standard cookie dialog, and Figure 17 shows one of the custom templates of the same CMP. The custom templates can also be further tweaked.

And even if a CMP could be easily detectable, many websites do not use a CMP and implement a cookie dialog any way they want. According to a study from 2020 [MBS20] only 6.2% of the 28,257 visited websites contained a known CMP while this substudy detected around 36% of the websites contained a cookie dialog.

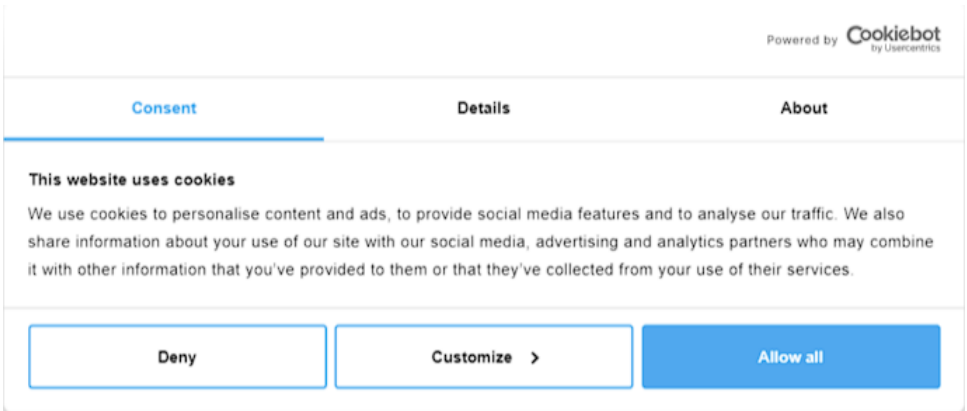


Figure 16: Standard cookie dialog from `https://www.cookieinfo.net/`

[26] `https://pypi.org/project/selenium-wire/`

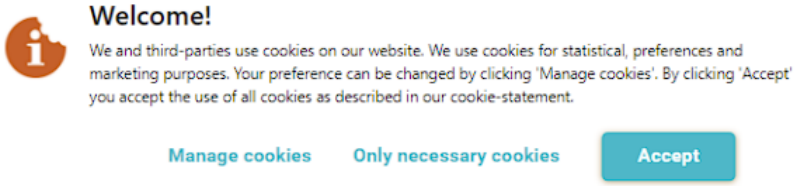


Figure 17: Custom cookie dialog from `https://www.cookieinfo.net/`

To alleviate this problem it is chosen to use certain HTML features to detect the presence of a cookie dialogs. Specifically iframes, z-indexes and active elements. These techniques are not solely used for cookie dialogs, but by using these techniques a list candidates is set up that can be a cookie dialog. The text inside all of the candidates is presented to a pre-trained machine learning model. This model predicts if any of the candidates is in fact a cookie dialog.

5. Detecting options in a cookie dialog
The options were taken into account for the cookie dialog: ACCEPT, DECLINE, MODIFY and SAVE. Everything that does not fit in any of these options is labeled as OTHER. Table 9 shows the declaration of each option. These options will be used by a machine learning model and by the crawler to simulate user interaction. Figure 18 shows a cookie dialog with the four different options in it. Some options on some cookie dialogs lead to a new screen or new website, but this was however not pursued in this study.

Options	Declaration
ACCEPT	Accept all cookies
DECLINE	Decline all cookies
MODIFY	Modify what cookies a user will accept or decline
SAVE	Save the settings for the selection of cookies
OTHER	Something unrelated to the above

Table 9: Classification of options

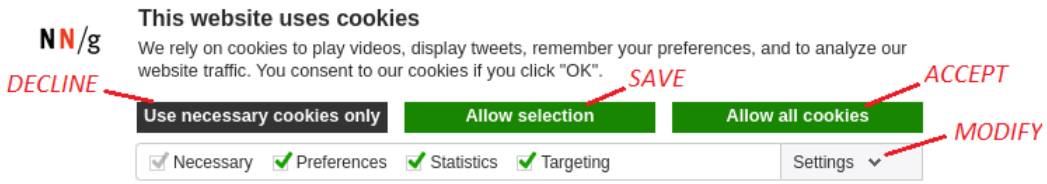


Figure 18: Cookie dialog from `https://www.nngroup.com/`

As previously mentioned, each and every website that uses a CMP can adjust the cookie dialog to its liking. The same applies to the options in a cookie dialog. Options can be added or removed, text can be altered, a different language can be used, the wording can be chosen to the liking of the website that implements them, and even the use of sliders or checkboxes can be chosen. (see Figure 16 and Figure 17 for two examples from

the same CMP). This results in making it very hard to hardcode information about the options even if the correct CMP is detected. If the website designs its own cookie dialog design hardcoded options cannot be used.

The different options can also be detected by hardcoded strings. This was originally used to detect the presence of the accept and decline options on a webpage. Detecting any of these options would then mean a cookie dialog is present. 39 English and 25 Dutch phrases were collected for ACCEPT options and used in a initial version of the crawler. This was however only workable for those two languages and only if it matches the exact string. Since each and every website can determine the wording of each option making exact strings hard to work with. The benefit of this option though was that the cookie dialog itself would not need to be detected, if an exact match of a string is found a cookie dialog would need to be present as well.

Eventually it was chosen to first detect a cookie dialog as previously mentioned. After a cookie dialog is detected all HTML elements are selected that can be used for an option. The text from all of these element are then fed into a pre-trained machine learning model. This model predicts the option an element corresponds to.

6. Machine learning models
Although hardcoded strings were an easy solution to detect the presence of a cookie dialog, this could not be scaled to a lot of websites because wording can be changed by each website and the list of websites will consist of around 30 different languages. For both the detection of a cookie dialog and its options, it is then chosen to use a machine learning model. A machine learning model is trained on a pre-classified list and then when feeding the model new text it can classify if the text belongs to a cookie dialog using multiple languages. Due to the limited practical knowledge of using machine learning models, a search was conducted to find an easy to work with implementation or library. A classification model was used by CookieEnforcer[KNHF22] which was a good starting point although there was no mention how this was practically used. A search for how this model can be used quickly arrived at Simple Transformers[27] an easy to use Python library that can train and use a model only three code lines. Although CookieEnforcer used an English only model, a better suited pre-trained model was found named XLM-RoBERTa that is capable of using multiple languages, which is used here.

7. Other supporting technologies
To save the cookies and the design information of cookie dialogs and its options to a persistent and reliable database, SQLite has been chosen because it is easily integrated into a Python program.

Multiprocessing was used to spawn multiple threads at the same time. This Python library simplified the start of a subprocess, the atomical usage of global variables, and the usage of a semaphore lock to prevent running too many CPU hungry executions at the same time and overloading the CPU.

SimpleGUI was used to display a simple window with user options and to show the current threads running, percentage CPU and RAM available.

8. Website list
Following a lot of previous studies [Aer21], [EEZ+15] and [KNHF22] for example and the

[27] `https://simpletransformers.ai/`

previous Section 4 it was decided to use the Tranco list to get a large list of websites. This list is a research based list of websites that puts the most visited websites at the top and less visited websites at the bottom and tries to reduce manipulation by websites for popularity. The order of the list will help determine if there are differences in cookie dialogs, options available or cookies being set between more and less visited websites.

9. Automated browser detection
Some websites detect the presence of an automated browser. If an automated browser is detected these websites can decide to show a different webpage not containing a cookie dialog where for a normal user a cookie dialog was present. Also different cookies can be set as a result. As this paper wants to show the user experience similar to reality some precautions were taken to prevent this from happening. Undetected-chromedriver[28] and Selenium-stealth[29] are two Selenium based alternatives for this problem. Although they both reduce the detection of an automated browser, during testing sometimes only 50% of the cookies were detected that could be found with basic Selenium. This would have skewed the results to show less cookies being set compared to a normal user.

Eventually the least detected solution was used as explained by Jonker et al. [JKV19]. The paper explains the details of preventing being detected as an automated browser. The best solutions was a standard Selenium running headfull with some additional settings.

5.3.2 PROOF-OF-CONCEPT IMPLEMENTATION

The crawler has been developed as a Proof of Concept and was extensively tested. Eventually the crawler has been used to analyse 100,000 websites. The results of this run has been discussed in Section 5.4

1. Environment Setup
All tests have been done on a home computer. It has an 8-core CPU capable of running 16 threads, has 64GB of RAM, an SSD hard disk and runs Windows 10 64-bit and Ubuntu 22.0.4 LTS. For further information on the programming environment, the modules used, and settings used for the browser see Appendix D.

2. Artifacts details
The following artifacts are used to execute the automated visits:

- *Populate database*: To populate the database with the tables and columns that are used and the websites to be visited
- *Cookie dialog extraction*: Given a browser session extract all possible cookie dialog candidates
- *Machine learning model for cookie dialogs*: To train the machine learning model using a classification of the cookie dialog candidates
- *Machine learning model for options*: To train the machine learning model using a classification of the options in a cookie dialog
- *Selenium*: To visit the website and interact with it
- *SQLite database*: To save all information about the visits

[28] `https://github.com/ultrafunkamsterdam/undetected-chromedriver`
[29] `https://pypi.org/project/selenium-stealth/`

- *Crawling*: To combine all artifacts into an automated, fast and stable environment

3. Populate database
The crawler uses an SQLite database where all information from each browser session is saved (except for the screenshots, they are saved to the harddisk). The not yet visited websites will also be prepared into this database so the crawler can read these websites to determine which websites are still to be visited. This database will thus first need to be filled with empty tables and columns. One of the tables is first filled with records of the list of websites and completed with empty columns.

The database consists of four tables. *elements* containing information from the elements in the website, *cookies* containing the information from the cookies, *predictions* containing all predictions made and *visits* containing all the visits that are made. The database setup is further discussed in Appendix C.

4. Candidate extraction
Candidate HTML elements for the cookie dialog need to be extracted from the website to be able to interact with them. The candidates will then be presented to the machine learning model to predict if it is actually a cookie dialog. Cookie dialog candidates are extracted using three different schemes: z-index, active element and iframe. The z-index setup has been used by Khandelwel et al. [KNHF22] to detect the topmost element of a webpage. This HTML technique is used to position an element above all other elements of the webpage. This makes the element with the highest z-index the first element to be viewed by the user, see Figure 19. The z-index extraction is used in two ways. First the highest z-indexes of the whole page and second the highest z-index of each HTML layer.

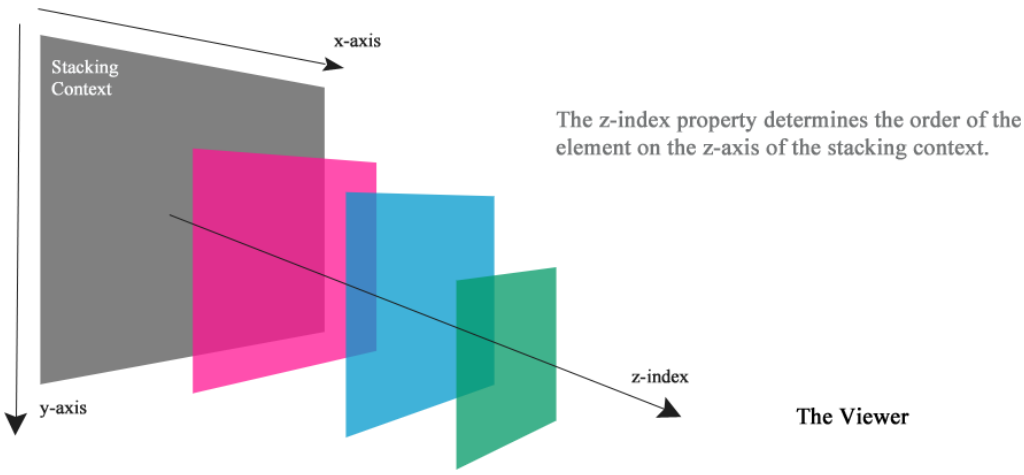


Figure 19: Z-index explanation from `https://tympanus.net/codrops/css_reference/z-index/`

The active element is usually the element that the browser session is focused on. If a cookie dialog is present there is a high chance the browser session is focused on this element. An iframe is a special element and it is sometimes used to show a cookie dialog by injecting the code into an iframe. An iframe needs to be entered by Selenium during the session otherwise the element cannot be used or interacted with.

At the end the candidates selection will then be finetuned and filtered. During finetuning it is tried to get as close to the element of the cookie dialog by traversing to the child if possible. During filtering candidates that cannot be a cookie dialog are discarded(style, ul, il, a, svg or button), candidates that have too little text (<50 letters), candidates that are the parent of another candidate, and candidates that are too small in physical size are discarded as well.

To extract the option candidate elements from a cookie dialog, HTML elements are extracted that can function as a button (button, a, span and svg).

5. Machine learning models
After selecting the candidates, it should be automatically detected if a cookie dialog candidate is in fact a cookie dialog and what function an option candidate has, two different machine learning models have been developed. A model is trained that can classify a given string of text. The machine learning models are based on the use in CookieEnforcer [KNHF22]. Initial settings have been re-used, but the model that has been used has been changed. CookieEnforcer uses a text classifier based on BERT.[30] BERT is a natural language model that has been trained using a large dataset of unclassified text. BERT can detect the contextual relations between words used in a text and is to be used as a starting point to train a model using a pre-classified list of data. BERT however is limited to the English language, to broaden the model to the use of other languages, a different model is used that is trained on multiple languages named XLM-RoBERTa.[31] This model allows training using only one or two languages and eventually doing predictions on all languages supported.

XLM-RoBERTa is used to develop two different machine learning models for use in the crawling artifact. The first model will predict if a cookie dialog candidate is in fact a cookie dialog. The second model will be able to classify the options in a cookie dialog. The library SimpleTransformers[32] is used to train the model and do predictions. This library is simple to use in Python requiring only a few lines to get started. Further information of the training of the different machine learning models can be found in Appendix E.

6. Automated browser detection
Techniques from Jonker et al. [JKV19] were used to limit detection of the presence of an automated browser. Further information about the automated browser and the settings used to prevent detection can be found in Appendix F

7. Crawling setup
Using the artifacts mentioned above, the main crawler commands the artifacts (see Figure 21 for an overview of the artifact). The crawler uses a *main thread* that calls in the different subthreads. The main thread first reads in the database if present in a list to visit. If this is a fresh restart the database will first be populated with the websites. The main thread starts the database thread, the GUI thread and browser sessions. The main thread checks for aliveness of the running subthreads and also restarts the threads when too many errors have occurred or a restart is requested. During the restart residual threads are killed and the temp folder is reset. This need to be done because with every Selenium session there is a possibility that the session has not exited gracefully and a lingering

[30] `https://huggingface.co/bert-base-uncased`
[31] `https://huggingface.co/docs/transformers/model_doc/xlm-roberta`
[32] `https://simpletransformers.ai/`

process remains that eats resources indefinitely and fills the temp folder. Lingering processes will start eating CPU resources and temporary files will fill the hard disk. Both reasons cause the system to block itself after awhile and further browser session all end up being unsuccessful. Unfortunately these errors do not result in a clear message, usually the website appears to not be reachable, and this only happens after visiting a larger amount of websites.

The *database thread* will run a writing thread every 90 seconds. The writing thread extracts the lines from the global variables and resets them. The lines are added to the correct database table. The database thread will stop immediately when a stop is signaled by the user After a stop is signaled either by the program, because the list of visited websites has ended, or by the user, because of too much failed visits, the database thread stops immediately. The writing thread however will be executed one last time after all threads have ended gracefully or stopped with a timeout or data will be lost.

The *GUI thread* will show 3 buttons to pause, unpause or stop crawling. Pausing can be helpful to temporarily free up bandwidth for other users in the network or if other actions need to be executed during the run, because while the browser sessions are starting and running they use a lot of bandwidth and they call focus from the OS making it hard to do something else.

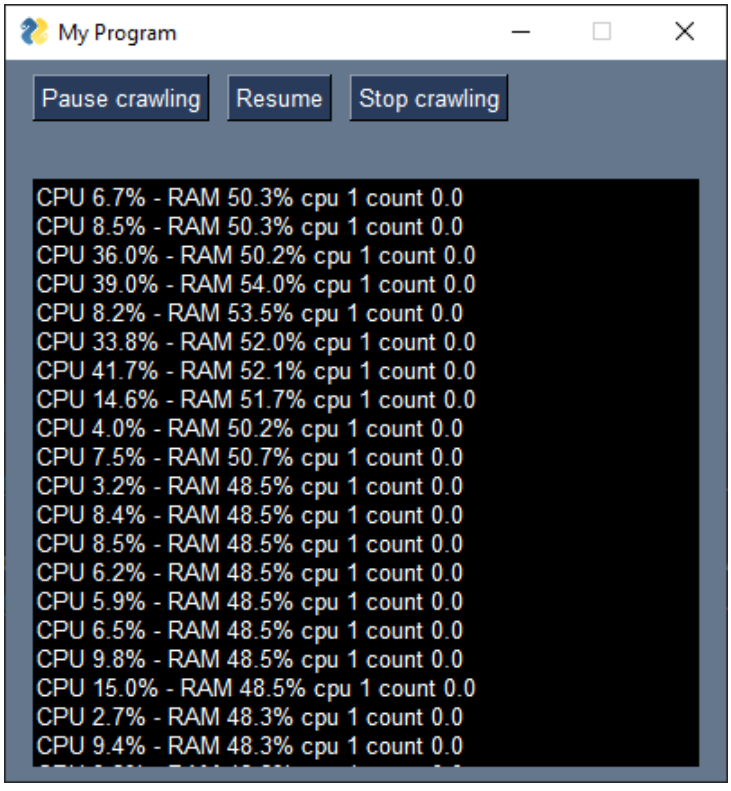


Figure 20: GUI thread

The *main thread* will launch as many *browser threads* as requested. These browser threads will retrieve the website that needs to visited, spawn a browser instance and monitor the instance. If a timeout occurs after 90 seconds, that would mean the browser has become unresponsive and needs to be killed or it will take up resources indefinitely. All children

of the thread will be killed off and a new browser instance is started with the next website. Killing an instance does however result in temporary files being left behind.

If it is an initial visit all cookies and a screenshot is saved and the cookie dialog candidates are prepared. The candidates are fed to the cookie dialog machine learning model. The first positively predicted cookie dialog is used further. After a positive detection information about the cookie dialog and a screenshot of it is saved. Then cookie dialog options from this cookie dialog are prepared and fed into the options machine learning model. The last occurrence of every individual type is to be saved. This is done because options are usually positioned at the end of the cookie dialog, and if multiple predictions of the same class is found then the last one has a higher chance of being the correct option. If an ACCEPT option is detected then the visit continues during this session and the ACCEPT option is clicked and all cookies and a screenshot is saved. Otherwise this session is finished. A screenshot after interaction can be used to check afterwards if the click has been successfully executed.

If it is a second visit then information of the option is extracted from the database. Using this information the option is detected and clicked on. After clicking on the option all cookies are saved and a screenshot is saved. If anywhere in this process an iframe is used for the cookie dialog, then the session needs to enter the iframe first before it can continue.

During all visits errors are detected and saved into the database to prevent a revisit of a failed website. Resources are split as much as possible. Global variables are directed by the multiprocessing library so only one thread can use each variable at the same time and racial problems are avoided.

For the machine learning models unfortunately using multiprocessing resulted in making it impossible to load the models once for the whole program and share the models between different threads, to minimize impact the models were loaded as late as possible by each thread if needed and unloaded as fast as possible. The predictions were made atomically to reduce resource spikes.

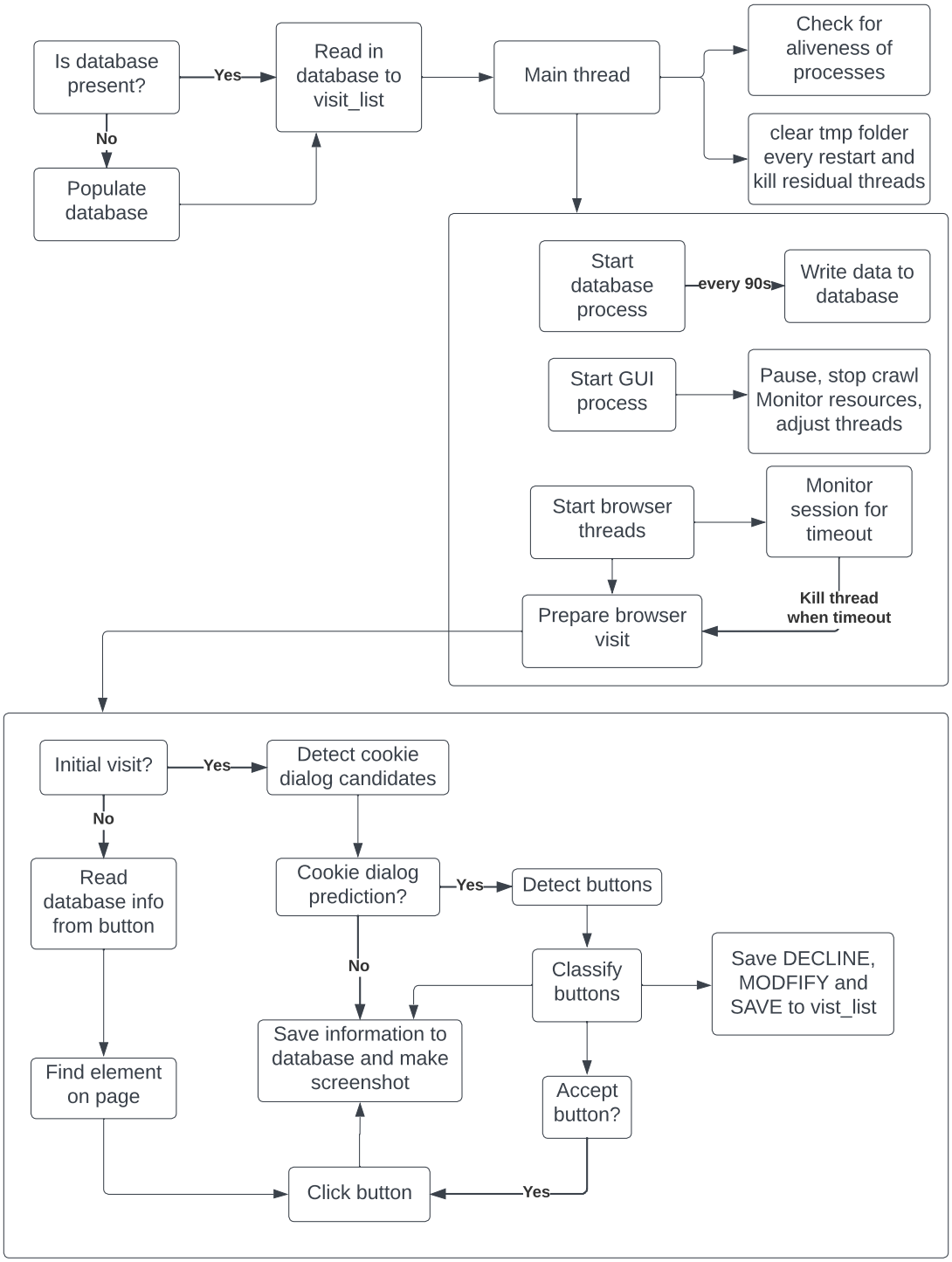


Figure 21: Workflow crawling

8. List of websites

The same 400 websites from the Section 4 are visited first. Added to this are the top 500 websites from each European ccTLD from the Tranco list. The same list was used in the previous paper [Aer21]. The following countries are used: (France (.fr); Ireland (.ie); The Netherlands (.nl); UK (.co.uk); Spain (.es); Belgium (.be); Austria (.at); Portugal (.pt); Ger-

many (.de); Romania (.ro); Luxembourg (.lu); Switzerland (.ch); Lithuania (.lt); Hungary (.hu); Sweden (.se); Slovenia (.si); Latvia (.lv); Italy (.it); Finland (.fi); Estonia (.ee); Czech Republic (.cz); Slovakia (.sk); Croatia (.hr); Poland (.pl); Norway (.no); Greece (.gr); Bulgaria (.bg)). These 27 countries will add 12,408 websites. Unfortunately not every country ccTLD has enough websites in the Tranco list.

This list is completed to a total of 100,000 websites. To get a representative look at the whole online happening the list of 1,000,000 websites from the Tranco list is divided into three parts (Table 10). The 25,000 most visited websites on the list are all visited, they have the biggest impact on the users. The remainder of the list could not be visited completely and is divided into two parts consisting of 75,000 and the remainding 900,000 websites. From the first part a random set of 30,000 websites are picked and from the last part a random set of 45,000 websites are picked. This scheme was used to get a balanced view of the 1,000,000 most visited websites.

Range start	Range end	Scheme	Percentage used
1	25,000	all websites	100%
25,001	100,000	Random 30,000	40%
100,000	1,000,000	Random 45,000	5%

Table 10: Crawler list composition scheme

10. Protection from malicious websites

Because a home workstation was used, it was decided to use some basic protection from malicious websites and adult entertainment. Malicious websites are filtered because they could inject a harmfull program that damages the hardware or changes the files on the harddisk. Because of the use of a separate OS to conduct the actual crawl this risk was minimized, but still present. Websites containnig adult entertainment are filtered because these websites are known to sometimes be malicious especially when they are less visited. Also because screenshots are taken the user did not want any adult or possibly illegal material saved to the harddrive. To accomplish this a DNS provider has been used during the crawl session that filters out DNS requests. The Family Shield from OpenDNS has been used.[33]

10. Testing

Testing the artifacts was done gradually and as much individually as possible during development. While building up the code, for each significant change in the code, the first few websites were visited and evaluated. To set up all basic functions small crawling sessions were conducted for up to 100 websites.

The gathering of the cookie dialog candidates was developed separately and tested until most of the 250 most visited websites were successfully detected. Eventually two cookie dialogs from this lest was not detected as a candidates.

The candidates of the 500 most visited websites were then used to train the machine learning model for the cookie dialogs. This model was then added and used to detect the cookie dialogs and then gather the candidates for the options in a cookie dialog. To train

[33] `https://www.opendns.com/setupguide/?url=familyshield`

the second machine learning model the same set of websites were used. A hardcoded list of strings was used to pre-classify as much of the options automatically as possible.

During testing several larger runs up to 2,500 websites were conducted as well. After testing several methods to execute the artifact in a Windows environment is was determined that for speed and security a Ubuntu environment would work the best. After larger tests of up to 25,000 websites more and more problems were resolved. Eventually the full run had to be redone multiple times.

The final run was conducted without any major problems during three sessions, because the workstation had to be used as well.

5.3.3 LIMITATIONS

The design choices and execution of them have resulted in some limitations.

- The automated browser detection mitigations could possibly not be thorough enough, which would mean a website could still detect its being visited by an automated browser and show a different website or set different cookies. Detection could also block the visit to a website entirely. This could change the results
- Some websites employ another layer of 'robot' detection in the form of a popup that needs to be clicked by a 'human' in a certain way to be able to advance into the homepage
- Some websites show another popup that blocks access to a cookie dialog, interaction of this popup was not taken into account
- The detection of a cookie dialog is not 100% reliable as is the detection of options in a cookie dialog. Which could change the results
- Although the artifact has been carfefully developed and extensively tested, it may not work exactly as intended. Errors could still arrive that make a website visit invalid.
- Using a DNS provider to shield of possible malicious and adult websites could change the results. Maybe these website are legitimate websites which are perfectly normal to visit?
- The set up of the machine learning model is prone to error. Did enough classes were used and were all classes labeled correctly?
- If multiple cookie dialogs are shown (which happened in 2 websites of the 500 most visited websites), only one is taken into account
- The crawler in its current design only interacts with the cookie dialog once. An option to decline all cookies where more than one interaction is necessary is currently not possible
- There still remain a few websites that could not be accessed and were not flagged as unreachable. These websites show a blank page with a notice the website could not be found

5.4 Results

5.4.1 Statistics of the crawling run

The final crawling run was done on 100,000 sites. The run was made between July 9, 2022 and July 12, 2022. These were the statistics of the final run:

- 100,000 sites visited
- Total runtime: 67 hours
- 2.4 TB of bandwith was used
- 136,949 screenshots were made totalling 68.8 GB of data
- The SQL file was 430MB large containing 2,521,353 records
- 16 threads used on average

These are the contents of the database file:

- 143,965 total visits of which 110,356 normal visits, 4,585 timeouts, 15,050 errors, 2,110 filtered by DNS, 312 bot detections by Cloudflare, 7,056 unreachable and 13,666 in an unreadable language
- 1,564,039 cookies were saved, with 138,535 unique names
- 245,089 predictions were made of which 134,934 options and 110,155 cookie dialogs
- 69,517 successfull initial visits were performed, of which on 44,378 websites no cookie dialog was detected and on 25,139 websites a cookie dialog was detected
- The machine learning model has classified 21,855 ACCEPT options , 6,577 DECLINE options, 13,332 MODIFY options and 2,201 SAVE options

5.4.2 DETECTION OF COOKIE DIALOG AND OPTIONS IN EUROPE

For all the graphs in this chapter, only European countries have been used. This will show the cookie dialog usage inside Europe, how many cookie dialogs are used with which technology, which options are available in a cookie dialog. From each European ccTLD up to 500 websites have been filtered from the Tranco list. Unfortunately not every country has enough websites in the Tranco list to be visited, or some websites could not be reached. Six countries have been discarded because of these reasons. Malta (.mt) (6), Bulgaria (.bg) (57), Greece (.gr) (95), Cyprus (.cy) (0), Luxembourg (.lu) (181) and Austria (.at) (157) have been discarded. Table 11 shows the number of successful visits for each country ccTLD.

Country	**visited websites**
Switzerland (.ch)	500
Latvia (.lv)	484
Estonia (.ee)	500
Slovenia (.si)	500
Norway (.no)	500
Hungary (.hu)	500
Lithuania (.lt)	500
Slovakia (.sk)	500
Czech Republic (.cz)	500
Portugal (.pt)	473
Finland (.fi)	468
Germany (.de)	457
The Netherlands (.nl)	500
Croatia (.hr)	500
Romania (.ro)	428
Belgium (.be)	396
Sweden (.se)	500
Poland (.pl)	476
Ireland (.ie)	468
UK (.co.uk)	498
Spain (.es)	446
Italy (.it)	497
France (.fr)	500

Table 11: List of European ccTLDs and the number of websites that have been successfully visited

Figure 22 shows the detected occurrences of cookie dialogs for the websites visited for each country ccTLD. The countries are ordered by the percentage of cookie dialogs detected. If a cookie dialog is detected then it is split in a normal cookie dialog and an iframe dialog. The graph shows there are big differences in the presence of a cookie dialog between the European countries. Most of the countries from Western Europe show more usage of a cookie dialog (France, Italy, Spain, ...), with The Netherlands and Germany being the biggest outliers. Countries that are from Eastern Europe tend to show less cookie dialogs. Switzerland is an outlier and shows the least cookie dialogs, they are not governed by EU laws and have implemented their own legislation. However if a visitor from EU visits a website from Switzerland, then the websites has to comply to EU rules. France is the biggest outlier that shows the most cookie dialogs, this is probably linked to the stronger rules imposed by Commission Nationale de l'Informatique et des Libertés (CNIL).

Looking at the technology used by a cookie dialog there are some countries that proportionally use more iframes than other, but there is no clear link between the countries. The choice between the usage of iframes or normal frames is probably arbitrarily done by a website. Most countries however only show minor usage of iframes.

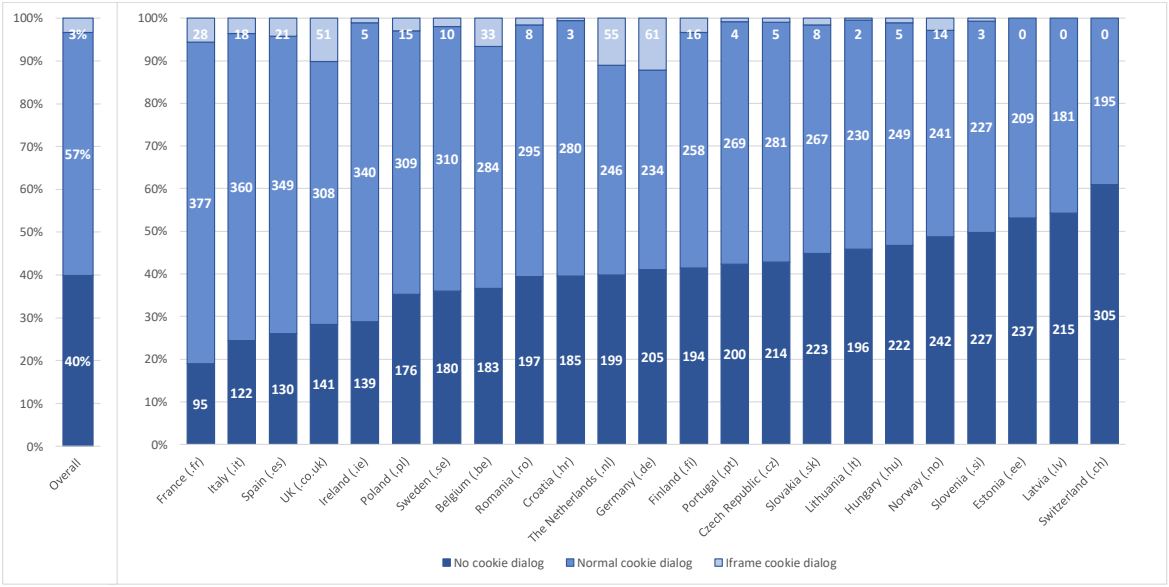


Figure 22: Occurrence of cookie dialog and the technology of it

Detection of options in European countries

Figure 23 shows for each European country the detection of ACCEPT and DECLINE options in a detected cookie dialog in percentages. The countries are ordered by the percentage of ACCEPT options detected. If an ACCEPT option is not detected then presumably the cookie dialog is only a notice and the only actual option is to close or ignore the cookie dialog without that action making a difference. A DECLINE options can only be detected by the crawler if it is present in the first cookie dialog screen that is shown. There is no clear line between which countries show an ACCEPT option, but if on average 85% of the European websites show an ACCEPT option then that would mean 15% of the websites show only a cookie notice that only tells the user cookies are used and the user is left with a 'comply or leave' option.

When looking at the DECLINE options, a much lower percentage than the ACCEPT option is detected here. On average a lot of websites do not use the DECLINE option in the first screen, either hiding this option in a different screen or not presenting a DECLINE option at all. France is the big outlier here with a big proportion of the cookie dialogs showing the DECLINE option. As was with the higher presence of a cookie dialog, this is probably linked to the tighter rules imposed by CNIL giving hefty fines for websites not implementing a clear DECLINE option. Facebook recently took a hefty fine for not implementing this option and eventually complied .[34] This shows that regulation does work, but there is still a lot of websites left in Europe that do not show a clear DECLINE option or do not give an option at all.

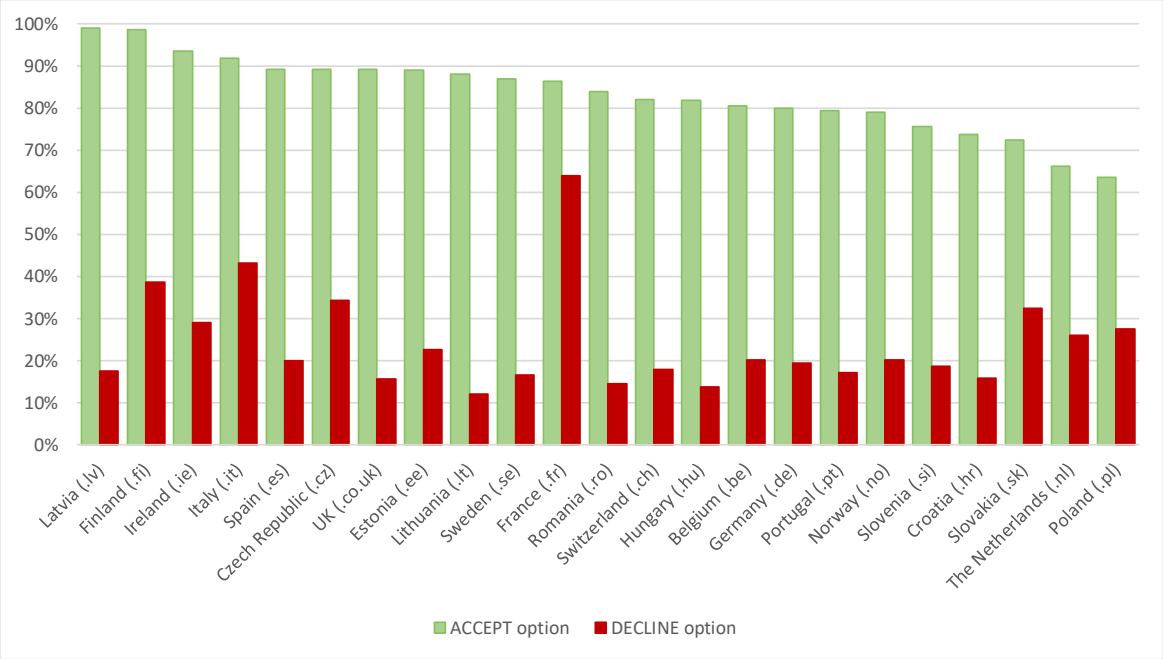


Figure 23: Detection of ACCEPT and DECLINE options in a cookie dialog

[34] https://www.huntonprivacyblog.com/2022/01/13/cnil-fines-big-tech-companies-210-million-euros-for-

5.4.3 DETECTION OF COOKIE DIALOG AND OPTIONS

To make all the graphs in this section the 100,000 websites that were selected from the Tranco list have been divided into 10 buckets containing an equal amount of successfully visited websites, totalling 69,503 websites. Table 12 shows the composition of the buckets. Because the Tranco list is ordered from most visited to least visited, these comparisons will show if there is a difference between the most and least visited websites. If there is a trend going from more visited to less visited websites, that will show if work needs to be done to regulate all websites and not only the more visited.

Range start	Range end	Number of websites
1	10373	6951
10374	20538	6951
20539	38718	6951
38719	63410	6951
63411	88356	6951
88357	170563	6951
170564	348090	6951
348091	547712	6951
547713	771411	6951
771412	1000000	6948

Table 12: Division in 10 buckets from the whole Tranco list

Detection of cookie dialogs

Figure 24 shows the detection of cookie dialogs in the different buckets. The trend in the graph is that the more visited websites show more cookie dialogs and lesser visited websites gradually show less cookie dialogs (The presence drops from 50% to around 22%). First possible explanation for this is that the more visited websites (have to) follow regulations better because regulating the most visited websites has the biggest impact on the users. Lesser visited websites do not always follow regulation to the letter because they are less looked at and the impact of regulating these websites is smaller. There are also a lot of websites that only stay alive for a relatively short period of time, and by the time these websites are looked at they are already no longer available. Second possibility is that less visited websites have less to gain by using more cookies and do not need to ask permission for using cookies if cookies that need permission are not used.

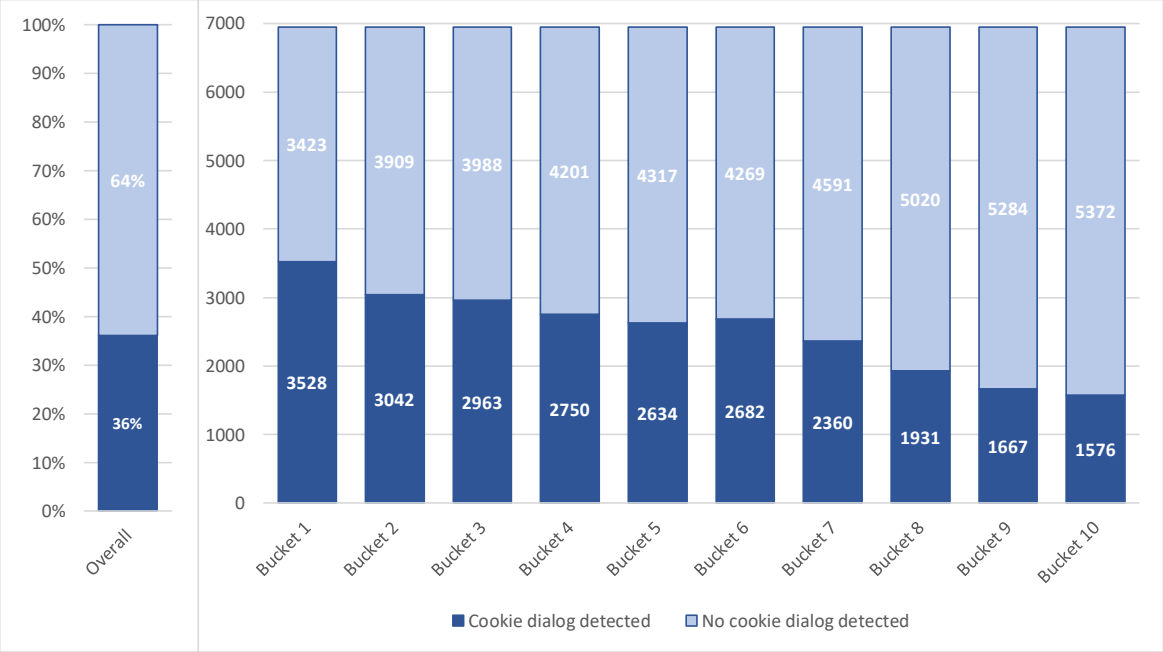


Figure 24: Detection of cookie dialogs in buckets

Options in a cookie dialog

Figure 25 shows the detection of the different options in a detected cookie dialog. The detection of an ACCEPT option goes down a little for the less visited websites. The trend of the DECLINE option and MODIFY option shows they are less detected on less visited websites. The graph also shows while a good amount of websites give the possibility of a MODIFY option, a DECLINE option is not present on a lot of websites. With less visited websites also showing less of those options than more visited websites. In an ideal situation all cookie dialogs should show the ACCEPT and DECLINE option. This graph shows that a great deal of websites are not implementing these options at the same rate, and less visited websites at even lower rates.

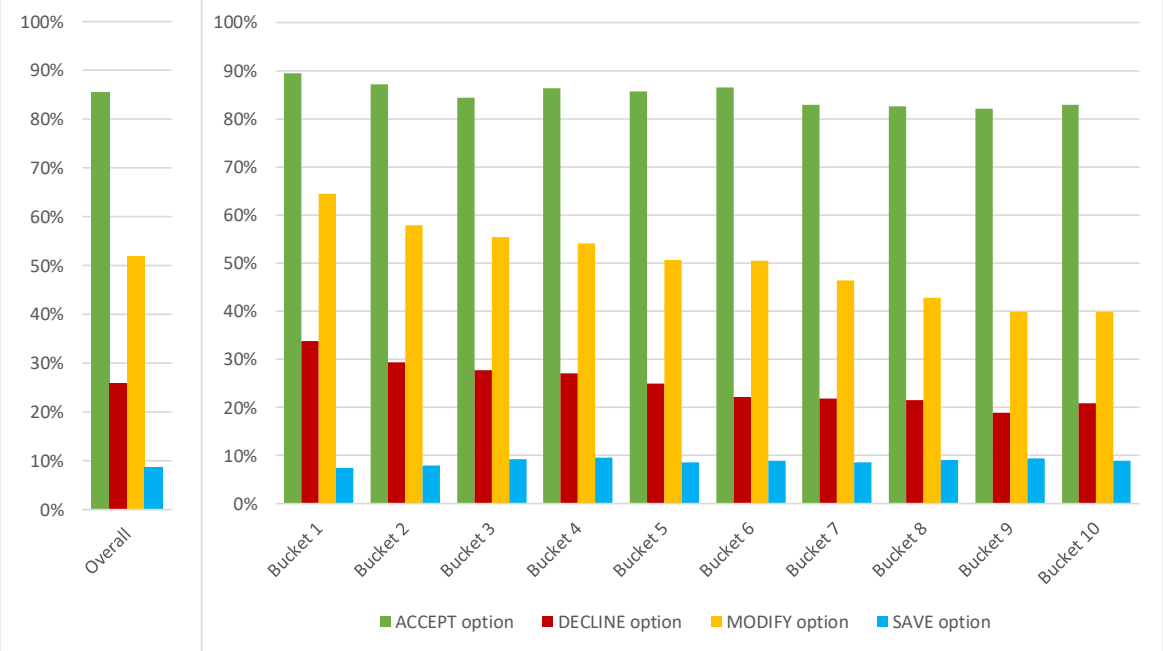


Figure 25: Detection of options in the detected cookie dialogs

5.4.4 COOKIES SET

Using the same bucket division, the graphs in this section will show the number of cookies being set with or without a cookie dialog and the number of cookies being set after interaction with any of the options. These will give an indication if the usage of the cookie dialog is related to setting cookies and if selecting options has an effect on the cookies begin set. The division in buckets will also show if more visited websites show different behaviour than less visited websites.

Average cookies set initially

Figure 26 shows the average cookies being set for each bucket if no cookie dialog is detected or initially before interaction with a cookie dialog. The graph shows on average for all buckets more cookies are being set if a cookie dialog is detected even though no action has been taken yet. This could be explained by the possibility of a cookie dialog setting one temporary cookie or more to indicate no consent has been given yet. And after giving consent this cookie can be reused or deleted depending on how the cookie(s) are used by the website. There is however a trend for less visited websites to set even more cookies with a cookie dialog present than without a cookie dialog present. On average 4 more cookies are set by less visited websites with a cookie dialog present. The question is if these cookies are legitimately used to set if no user action has taken place yet for example or are websites setting more cookies even though a cookie dialog is present that needs interaction?

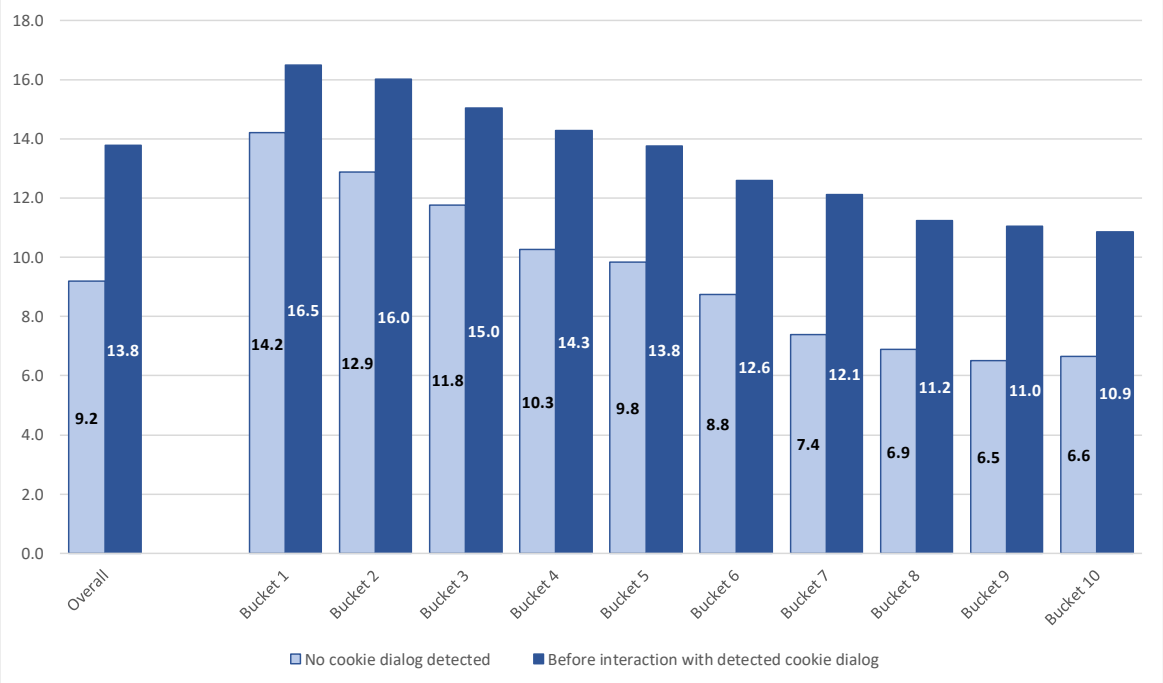


Figure 26: Average initial cookies set with or without a cookie dialog present

Average cookies set after interaction

Figure 27 shows the average cookies being set for each bucket after interacting with a cookie dialog. As should be expected most cookies are set after the crawler interacts with a detected ACCEPT option. The trend shows that more visited websites use more cookies than less visited websites. This could be explained because more visited websites can use or sell more information from their visitors or can use more external advertisers to collect data. Less visited websites could have less attention from advertisers because of the less visitors, thus using less cookies from advertisers.

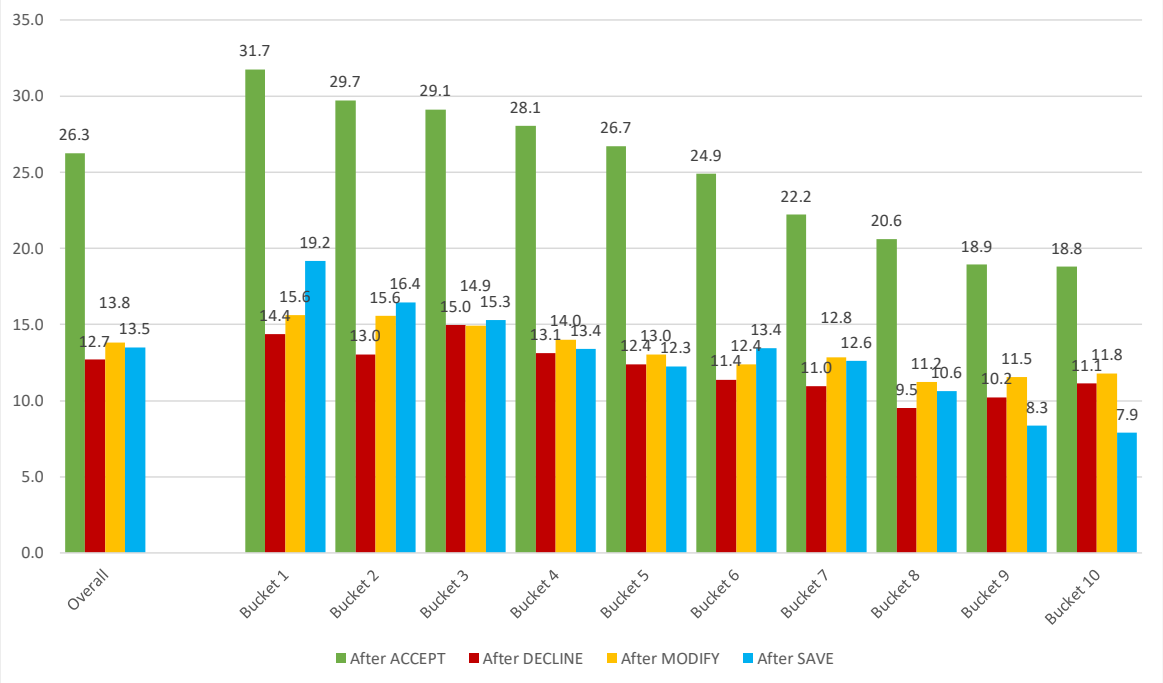


Figure 27: Average cookies set afrter interaction with a cookie dialog

In the next three graphs only websites are taken into account where a cookie dialog is detect with both the ACCEPT and DECLINE option. This is done to paint a picture of how many extra (or less) cookies are being set after interacting with the cookie dialog and specifically with the ACCEPT and DECLINE option. On a total of 5,519 websites both of these options were detected.

Extra cookies set after ACCEPT option

Figure 28 shows the number of websites that set a given number of extra (or less) cookies, after interacting with the ACCEPT option. Going right on the graph would mean more cookies are set after accepting, going up would mean more websites set this specific amount of extra (or less) cookies after accepting. As is to be expected most websites set more cookies after accepting the use of cookies, this is normal behaviour. There are however a lot of websites that set a lot more. One of the website (voetbal24.be) even sets an unbelievable amount of 436 extra cookies after accepting.(As a sidenote 30 of those cookies are first party and the other 406 are third party cookies). Off course this is an outlier and the median is only 6 extra cookies set after accepting. Against all logic there are a total of 39 websites that remove cookies after accepting the use of cookies. Three websites even remove 6 or more websites. Why a website would remove cookies after the user confirms the usage of cookies is incomprehensible.

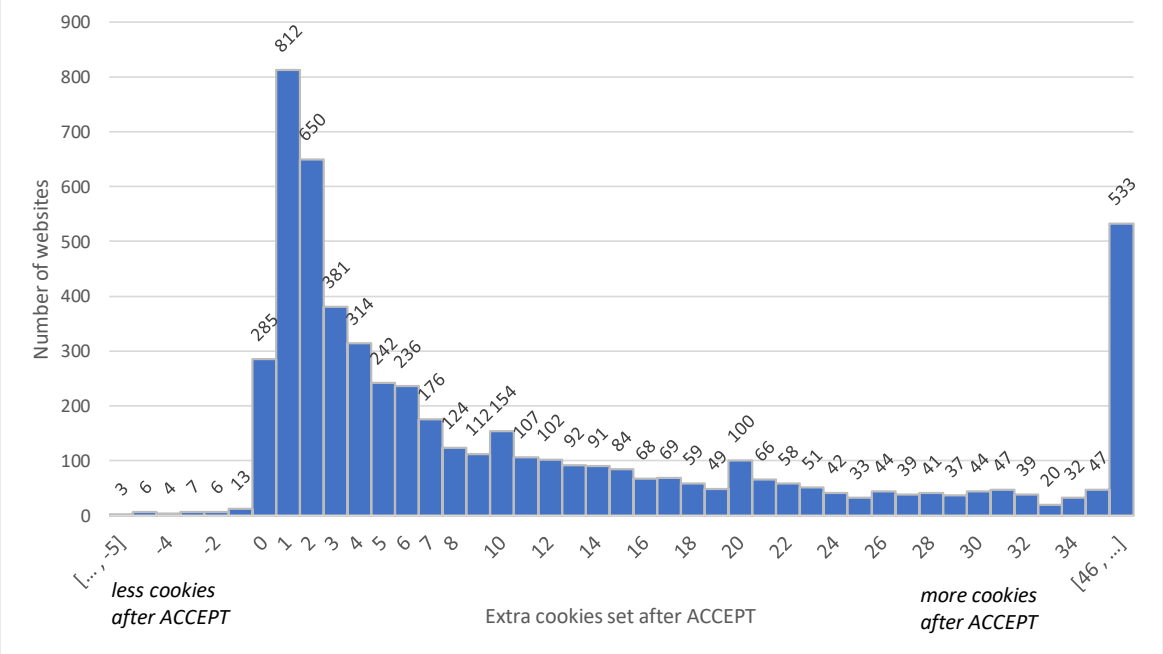


Figure 28: Number of websites that set extra cookies after interacting with the ACCEPT option

Extra cookies set after DECLINE option

Figure 29 shows the number of websites that set a given number of extra (or less) cookies, after interacting with the DECLINE option. The difference between this and the previous chart is here the DECLINE option is taken into account instead of the ACCEPT option. Opposite to the ACCEPT option, after interacting with the DECLINE option one would expect cookies being removed or staying the same. Cookies do not need to be deleted if one of the cookies is used as a placeholder to set the cookie preference. Even adding one cookie can be accepted if the meaning of that cookie is 'the user has declined using cookies'. It is however highly unexpected that a great deal of the websites set two or more additional cookies after declining using cookies (2,163 of 5,519 = 39%). The median however remains one cookie extra set after declining consent. The highest amount of extra cookies set was 181.[35] The highest reduction in cookies was 58.[36] It is to be noted though that the crawler uses two different visits to assess the cookies set during an initial visit and the cookies set after interacting with the DECLINE option.

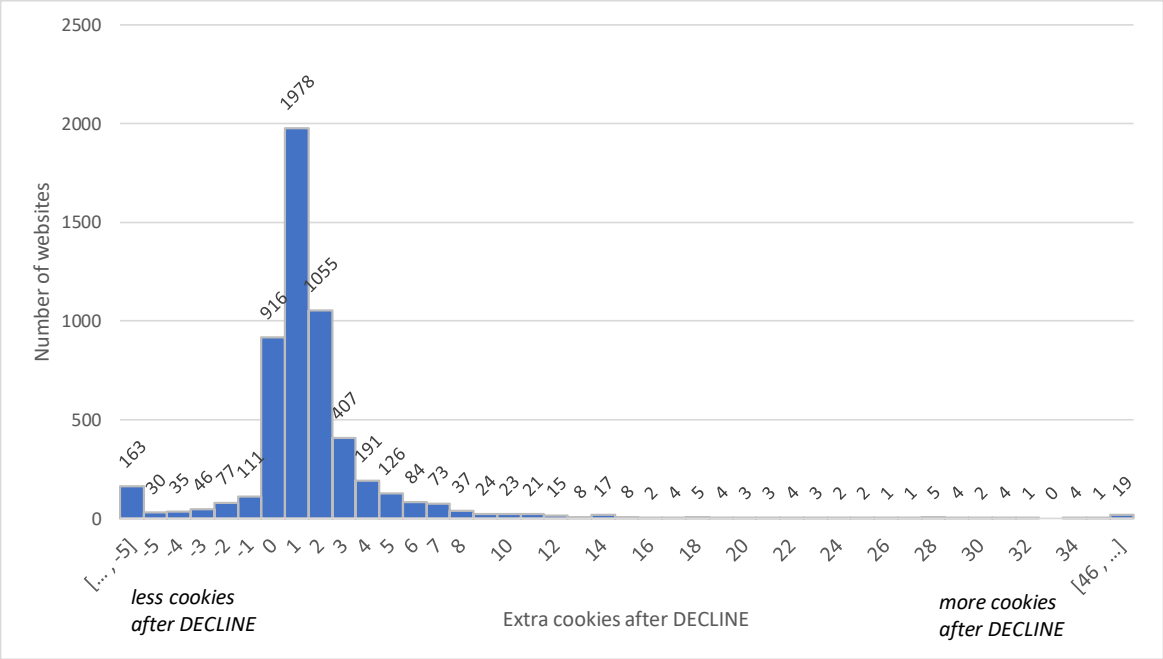


Figure 29: Number of websites that set extra cookies after interacting with the DECLINE option

[35] http://www.donald.pl

[36] http://www.telegrafi.com

Cookies set after ACCEPT or after DECLINE

Figure 30 shows the number of websites that set a given number of extra (or less) cookies, after interacting with the ACCEPT option compared to the DECLINE option. A positive number means more cookies are set after ACCEPT than after DECLINE. The two preceding graphs already show what would be expected here. A lot of websites set more cookies after accepting the use of cookies than declining the use of cookies. Just like before, something unexpected here is that there are a good number of websites that set less cookies after accepting the use of cookies than after declining cookies. A good explanation as to why this would happen was not found. The only thing that could affect the numbers here is that both visits are independent from each other. Although these visits were done relatively close to each other there could be irregularities in these websites that show a different number of cookies than expected.

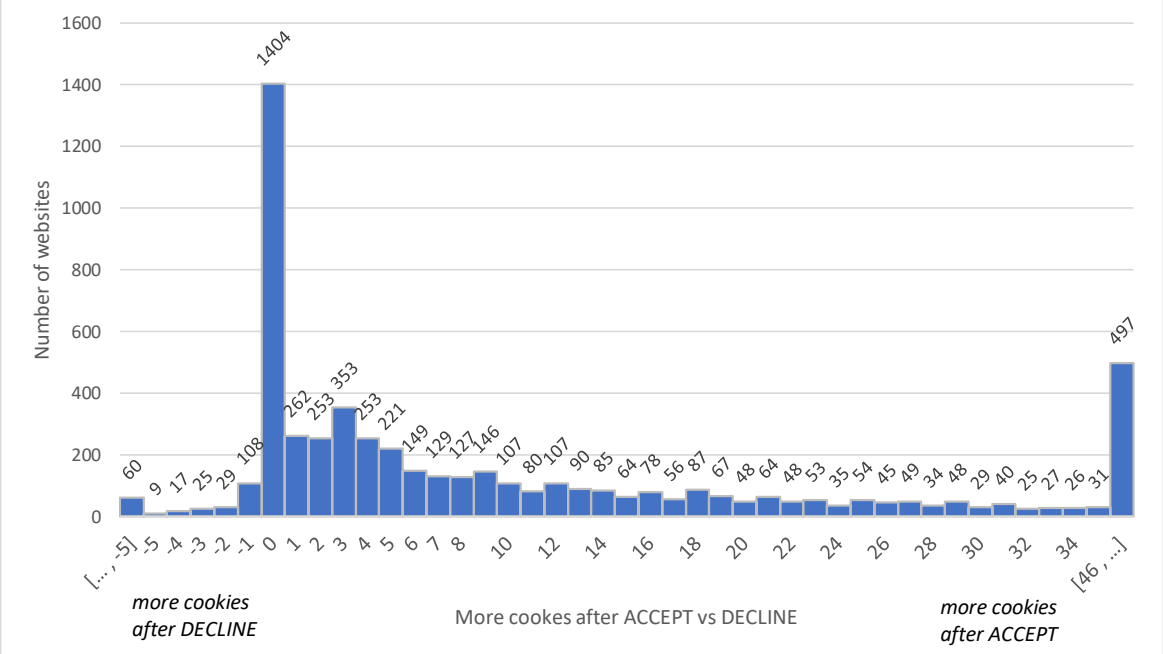


Figure 30: Number of websites that set a certain difference in cookies after interacting with the ACCEPT option vs the DECLINE option. Positive numbers mean more cookies are set after ACCEPT then after DECLINE

5.4.5 First and third party cookies

Websites can set two kinds of cookies. First party cookies and third party cookies. First party cookies originate from the domain of the website, where third party cookies originate from another domain. Most browsers are fazing out the usage of third party cookies, block them by default. Unfortunately Chrome being one of the most used browsers (it currently has a marketshare of 65%)[37] still allows third party cookies by default (they can be blocked by the user though), they were initially set to be blocked in 2020[38] and the current prognoses is that they will eventually be blocked by default in 2024.[39] Even though third party cookies will be blocked by default eventually by all browser, they are still extensively used. The following graphs show details about the third party cookies.

Third party cookies

Figure 31 shows the average number of first and third party cookies for each bucket that are being set no matter if a cookie dialog is detected or not. The results between a detected cookie dialog and no cookie dialog differ a little but the overall trend remains the same. Not shown in the graph, but a little more first party cookies are set without a cookie dialog and the amount of third party cookies stays roughly the same. Like previously the trend shows lesser visited websites set fewer cookies, the distinction between first and third party cookies show that on average there is percentage wise an equal reduction in first and third party cookies. Over all buckets the ratio between first and third party cookies stays roughly the same, at around half the amount of third party cookies are set compared to the first party cookies.

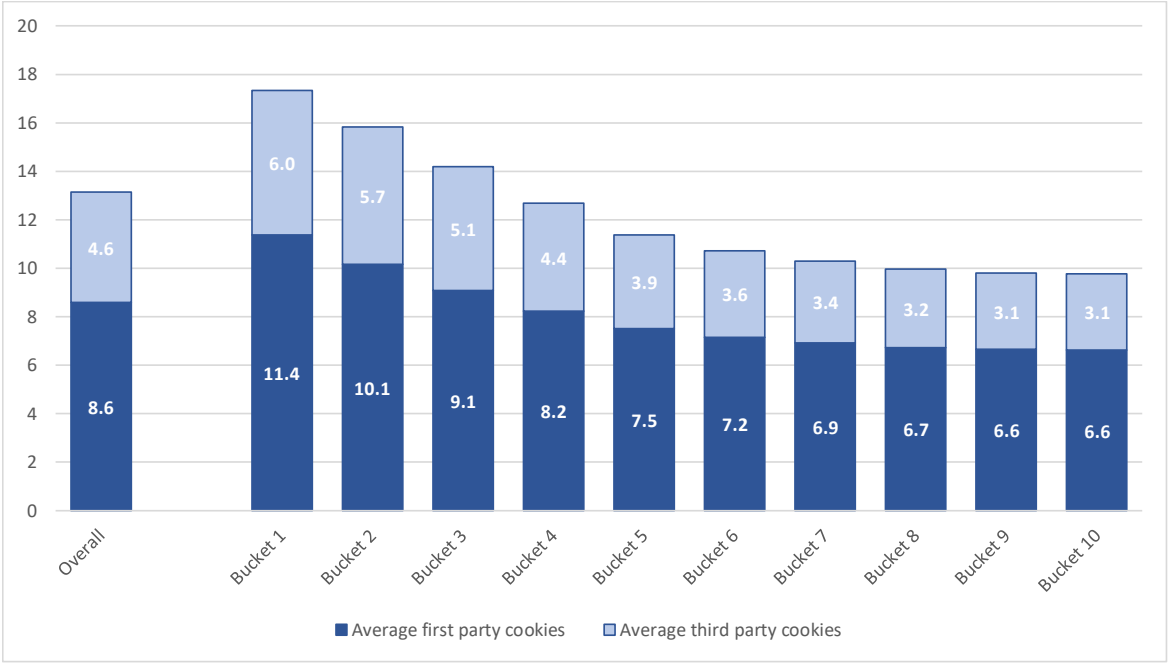


Figure 31: The average number of first and third party cookies set

[37] https://gs.statcounter.com/browser-market-share
[38] https://blog.chromium.org/2020/01/building-more-private-web-path-towards.html
[39] https://blog.google/products/chrome/update-testing-privacy-sandbox-web/

Figure 32 shows the median instead of the average. As can be seen the median is much lower, this shows that half of the websites only set one or two third party cookies. Because of the lower median number here compared to the average shows that websites that do use first and/or third party cookies tend to use a great deal more cookies.

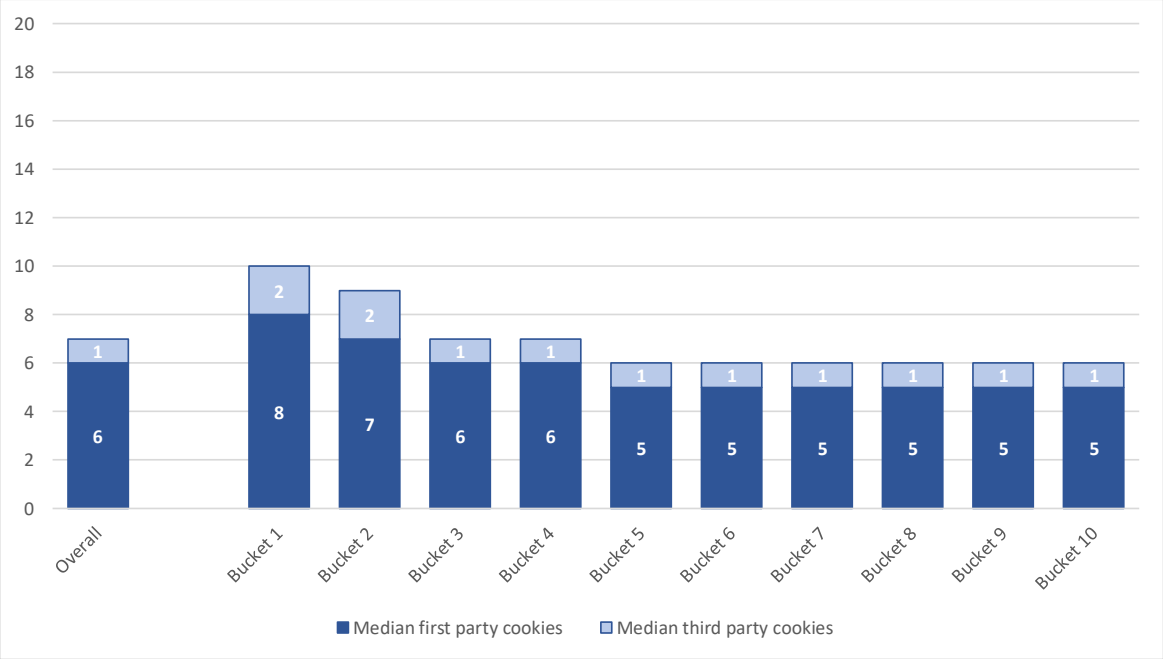


Figure 32: The median number of first and third party cookies set

Third party providers

Figure 33 shows for the 10 most used third party cookie domains on how many websites a third party cookie was set from this domain. A distinction is made before and after interacting with a cookie dialog, and if no cookie dialog is detected. The domains are ordered according to the appearance before consent is given. As was to be expected third party cookies are used by big advertisers: doubleclick (part of google), bing (Microsoft), google, clarity (Microsoft) and youtube. Adnxs and nr-data are lesser known advertisers. All of the companies from these domains invest a lot of money into advertising and are probably using cookies to collect data from the users on other websites. This should be perfectly fine if these cookies are set after consent is given. The graph does indeed show that after giving consent on average these domains appear on double the amount of websites than before consent is given. The question however is: why would cookies from these domains be used before consent is given? Looking at the cookies there are a lot of cookies named 'test-cookie' that has the value 'CheckForPermission' and expire in 15 minutes. The use of these cookies is acceptable, however an equal amount of cookies have the name 'IDE' and set a unique identifier in the value of the cookie. These cookies are obviously used for tracking purposes. Equally disturbing is the amount of websites third party cookies from these domains are present on without the presence of a cookie dialog. What is the legitimate reason so many websites set cookies from these domains without consent or without a cookie dialog?

The numbers on itself are also eye opening. The biggest domain doubleclick has at least one cookie on 25% of all visited websites before consent is given, after consent on a staggering amount of 53% of the websites, but even without a cookie dialog the domain is present on 22% of the websites. The big question is why are these domains setting cookies without consent is given. One of the possibilities is that these domains are injecting cookies without the user permission when an ad is loaded.

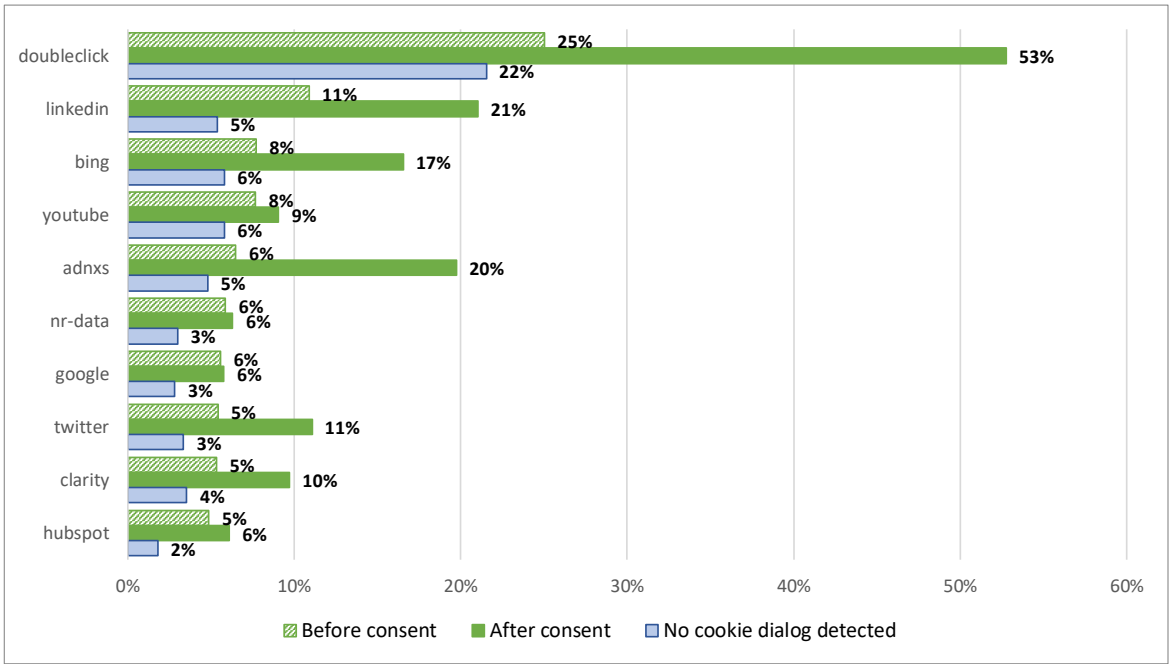


Figure 33: The the appearance percentage of the 10 most used third party cookie domains before or after interacting with a cookie dialog or if no cookie dialog is detected

Cookies set by third party providers

Figure 34 shows the average number of cookies set by each third party provider, if that provider sets at least one cookie on that website. Most domains only need to set one or two cookies, but linkedin for example uses on average 7 to 9 cookies on each website it is detected on. Surprisingly before giving consent (if that option is available) most providers on average set more cookies than after consent.

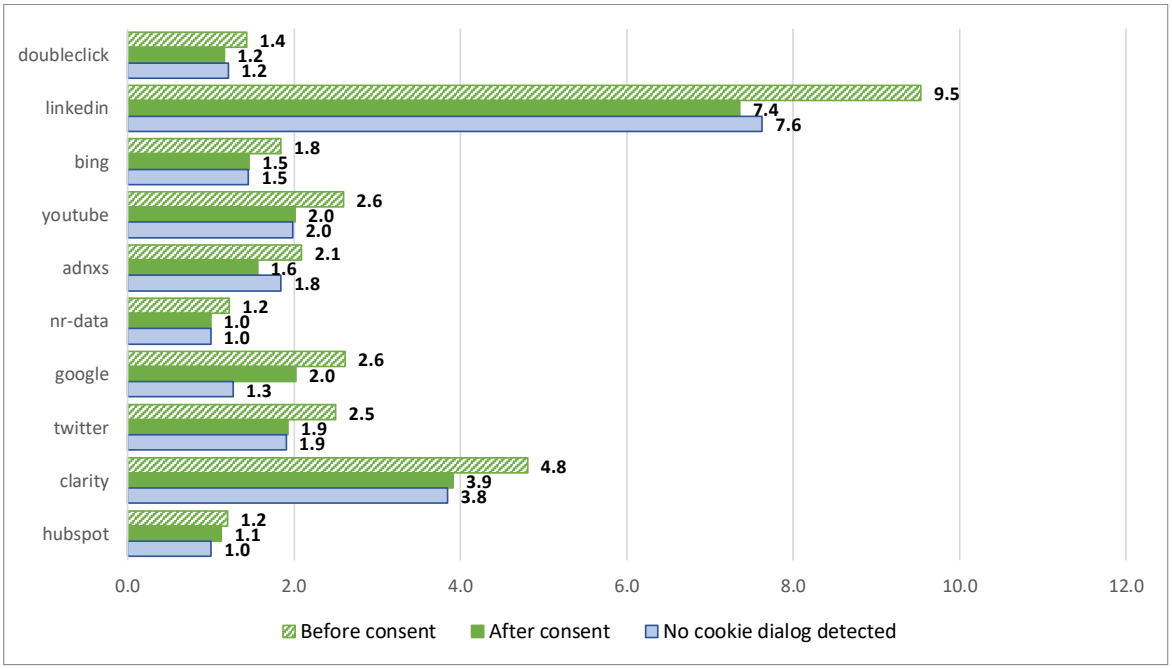


Figure 34: The average number of cookies set by each third party provider before or after interacting with a cookie dialog or if no cookie dialog is detected

5.4.6 HTTPS REDIRECTS

It is safe to say security is important when visiting websites. Not only for protecting the users computer and data on it and the login details of the user, but also for privacy reasons. The correct use of a cookie dialog is important, but if the connection between the user and server is insecure, attackers could inject scripts, images or ad content without the user knowing. To overcome this problem the HTTPS protocol was introduced: a secure version of HTTP using encryption and obfuscation of data. Using this secure protocol the user is protected from attackers.[40] A website can redirect a user from HTTP to HTTPS when visiting a website to make sure a secure connection is established. Englehardt et al. [EN16] did an extensive review of HTTPS redirects in 2016. The paper suggested that the use of insecure third party cookies impeded the adoption of HTTPS by websites. Only 8.6% of the websites visited in 2016 always redirected to HTTPS. Using HTTPS has become almost mandatory though, even if a website doesn't use sensitive data, an attacker can still inject scripts, images or ad content that can infect the users computer of retrieve sensitive data. There are however still enough websites that do not implement HTTPS. Nowadays browser tend to show a warning if an insecure connection is used. Figure 35 shows the warning that is given by Chrome when visiting an insecure website.

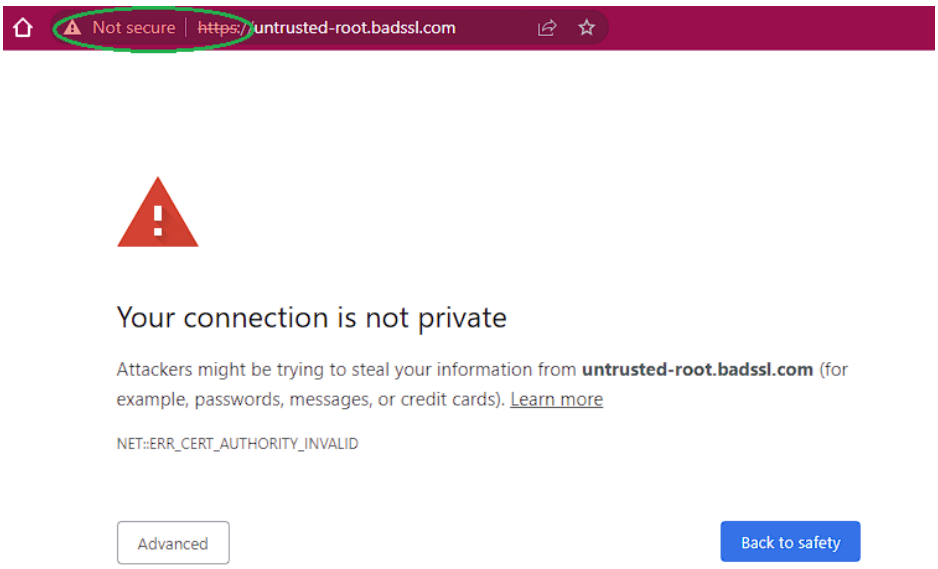


Figure 35: Example of insecure connection

[40] https://doesmysiteneedhttps.com/

Redirection to HTTPS

Figure 36 shows, during the crawl of this study, how many websites stay on HTTP and how many redirect the visit automatically to HTTPS. During this crawl 83% websites redirected to HTTPS, a great deal more than the detected 8.6% in 2016. The trend shows that the more visited websites tend to use more HTTPS on average. This could be explained by lesser visited websites using less sensitive data, and do not see the need to use encrypted data even though it is needed to protect the user. Compared to 2016, most website have started using HTTPS to secure the connection, however a little push is needed to get the last websites covered as well.

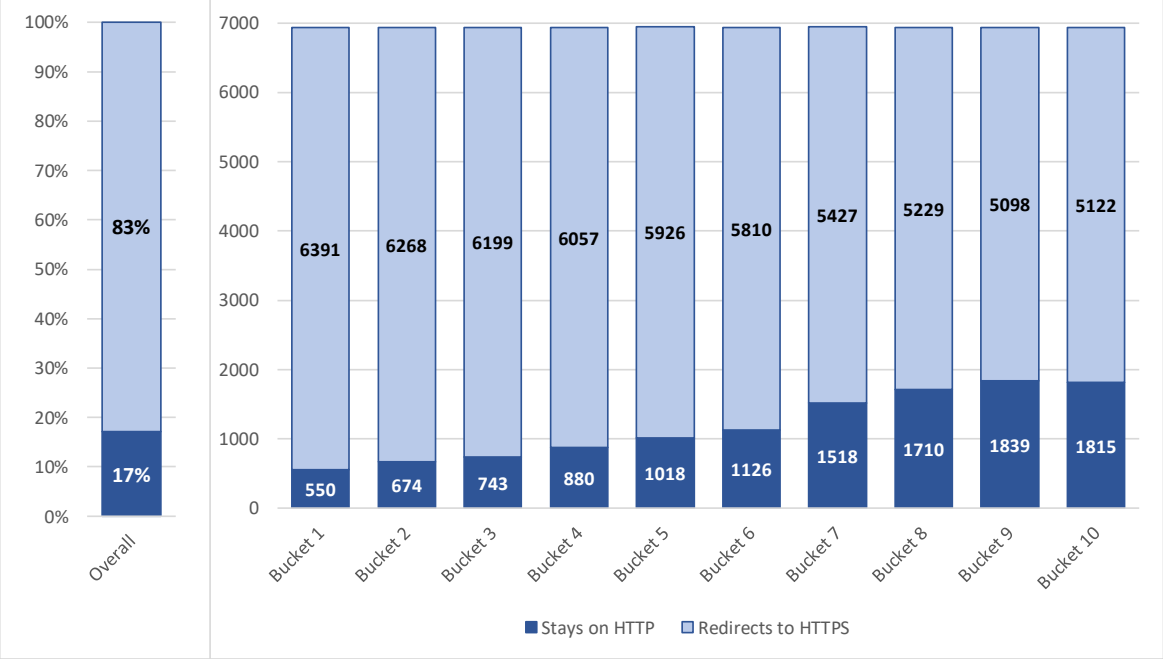


Figure 36: How many websites stay on HTTP and how many redirect to HTTPS

Usage of secure flag and httpOnly flag in cookies

Using HTTPS is good practice, and if HTTPS is used cookies are sent encrypted as well. There are however two settings used by cookies that can secure them as well. Otherwise an attacker may still use unencrypted measures to steal cookies as well. If an attacker, for example, can steal a cookie that stores the identification of a user on a certain website, the attacker could possibly login on that website and act as a user.

The *secure* flag makes sure the cookie cannot be used from an insecure HTTP connection, making sure the cookie is always sent encrypted to a server. The *httpOnly* flag makes sure the cookie cannot be accessed directly from the browser, making it impossible for an attacker to inject Javascript and read the cookie.

Figure 37 shows for the top 10 third party domains the usage of the secure flag and the httpOnly flag. The secure flag is used by almost all cookies from all big providers in any situation. For the lesser known providers however the usage of the secure flag drops significantly. The httpOnly flag is used much less, with the exception of a few big providers. This leaves room for an attacker to steal a cookie. No big differences are seen between the usage of a cookie dialog or before or after consent. Making this mandatory should make the use of cookies safer. There is a possibility that non sensitive cookies do not use secure measures, but it would still be good practice to always use these secure settings. Why would these measures not be used for every cookie?

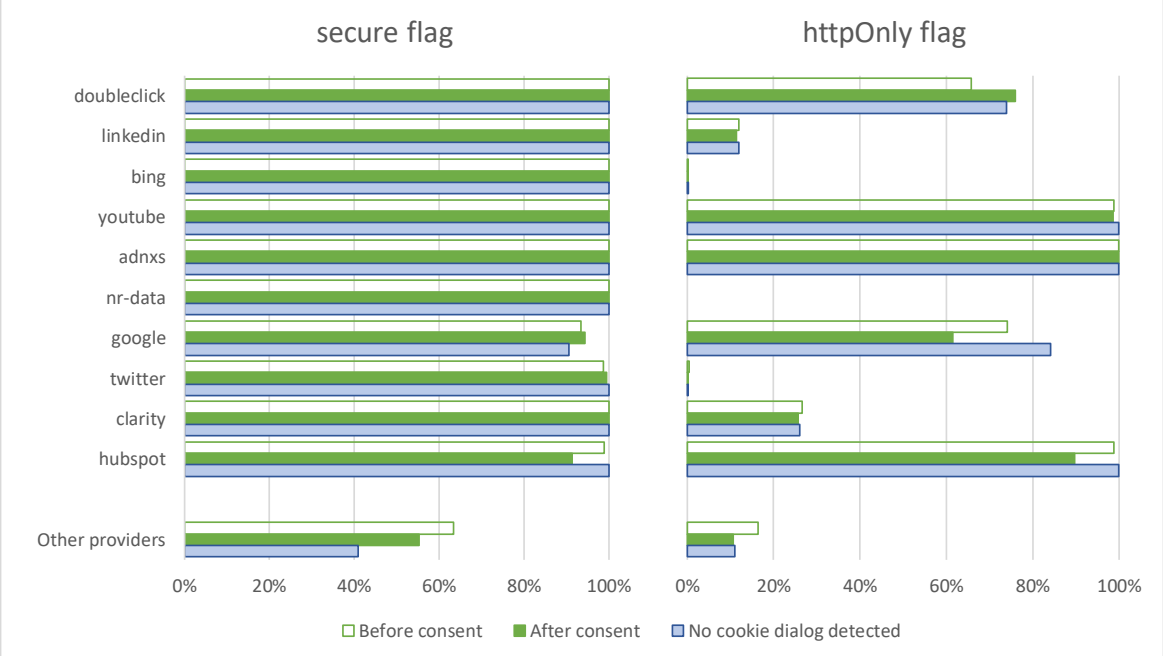


Figure 37: How many cookies from each domain use the secure flag and the httpOnly flag

5.4.7 Comparison to Section 4

The results from the crawl are compared to the manual browsing sessions from Section 4. Using the same 400 websites divided into the same buckets. The result will enable a brief assessment if data collected here corresponds to manual visits. The detection of cookies should be comparable, especially if the same options are detected and interacted with. The crawler however was not able to go through a cookie dialog in the same way a user could. The crawler only takes the options in the first window of a cookie dialog into consideration, and a manual visit can go though all options that are displayed. And although it is understood the crawler will not catch all exceptions to a rule and will not be able to detect all cookie dialogs and options, manual detection of options is also prone to error. A user might miss an option or cookie dialog that is present.

Detection of cookie dialogs

Figure 38 shows the detection of cookie dialogs during manual visit and in the crawling session. It highlights if no cookie dialog is detected during both, if a cookie dialog is not detected by both, or by only one. Overall in a small amount of websites a cookie dialog was not detected by the crawler. In the less visited websites the crawler misses more cookie dialogs compared to the manual detection of a cookie dialog. Overall on 5.8% of the websites a cookie dialog was not detected by the crawler vs manual. The high number can, besides being less detected by the crawler, also be explained by not using iframes in this selection of websites. Iframes could be easiers detected by the crawler. On the other hand on a few occasions the crawler detected a cookie dialog, while the manual crawl did not. This can be due to the cookie dialog being very small, staying undetected by the user.

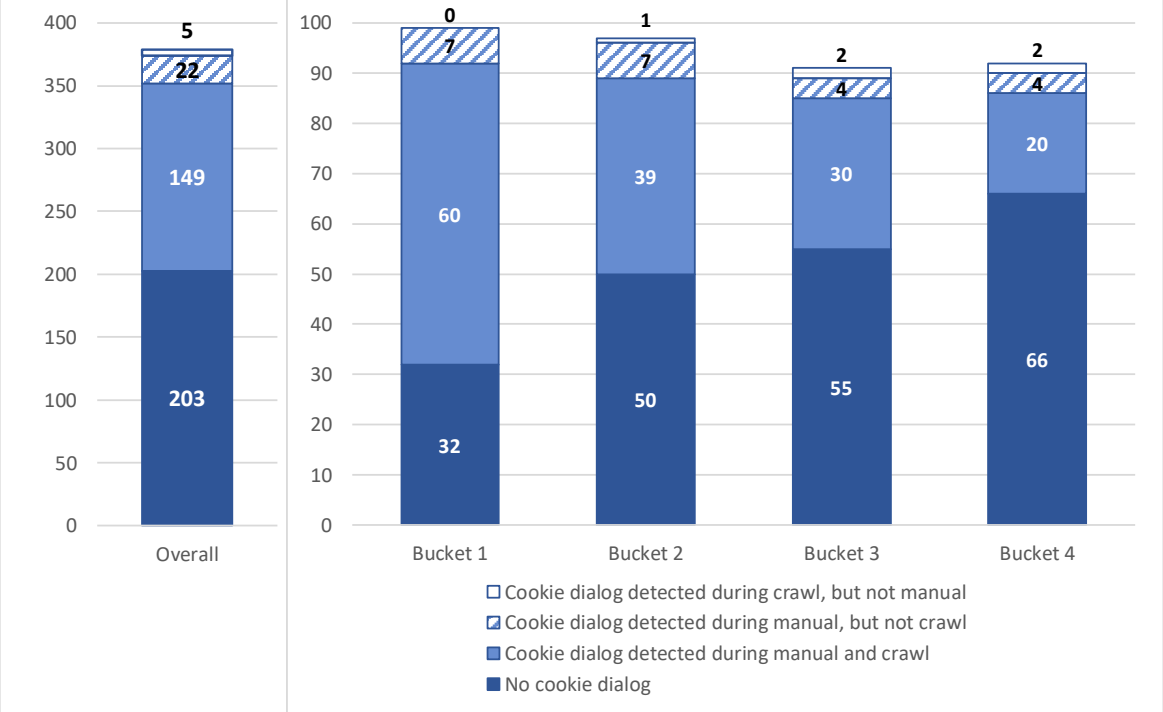


Figure 38: Detection of cookie dialogs during manual visit and in the crawling session

Detection of ACCEPT option in cookie dialog

Figure 39 shows the detection of an ACCEPT option in a detected cookie dialog during manual visit and a detected cookie dialog in the crawling session. It highlights if no ACCEPT option is detected during both, if an ACCEPT option is detected by both, or by only one. It has to mentioned that the crawler already detected less cookie dialogs compared to the manual detection. Overall the same can be seen as the previous graph: some ACCEPT options were not detected during the crawling session, but were detected during a manual visit. In all the buckets the crawler misses the same amount of ACCEPT options. If the previous numbers are taking into account that less cookie dialogs were detected, then only a few ACCEPT options were not detected during the crawl if a cookie dialog was already detected. Only 8 ACCEPT options were not detected if a cookie dialog was already detected. There could also be problems with interacting with the cookie dialog or the options were differently classified by different users. An ACCEPT option could also be interpreted as a 'close cookie dialog window'. Here as well the crawler detected a few ACCEPT options that were not detected during a manual crawl, this likely because of a wrong classification by the options machine learning model.

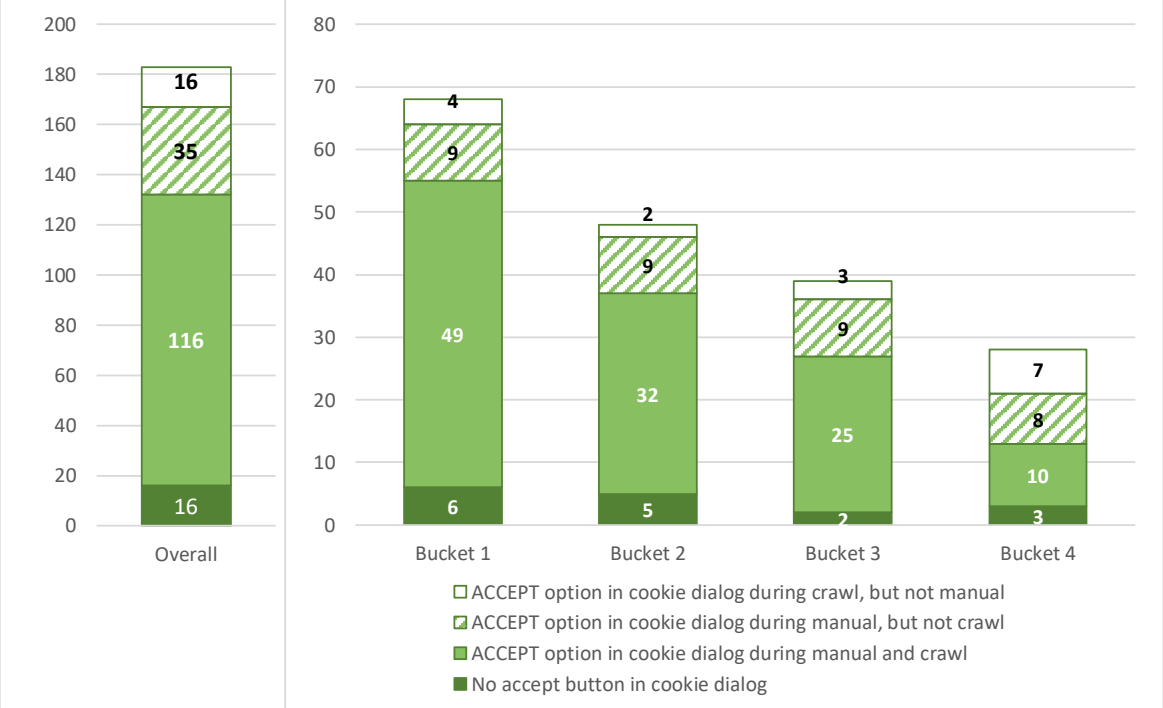


Figure 39: Detection of ACCEPT option during manual visit and in the crawling session

Detection of DECLINE option in cookie dialog

Figure 40 shows the detection of a DECLINE option in a cookie dialog during manual visit and in the crawling session. It highlights if no DECLINE option is detected during both, if a DECLINE options is detected by both, or by only one. According to the data the crawler does a worse job in detecting a DECLINE option in the first two buckets.

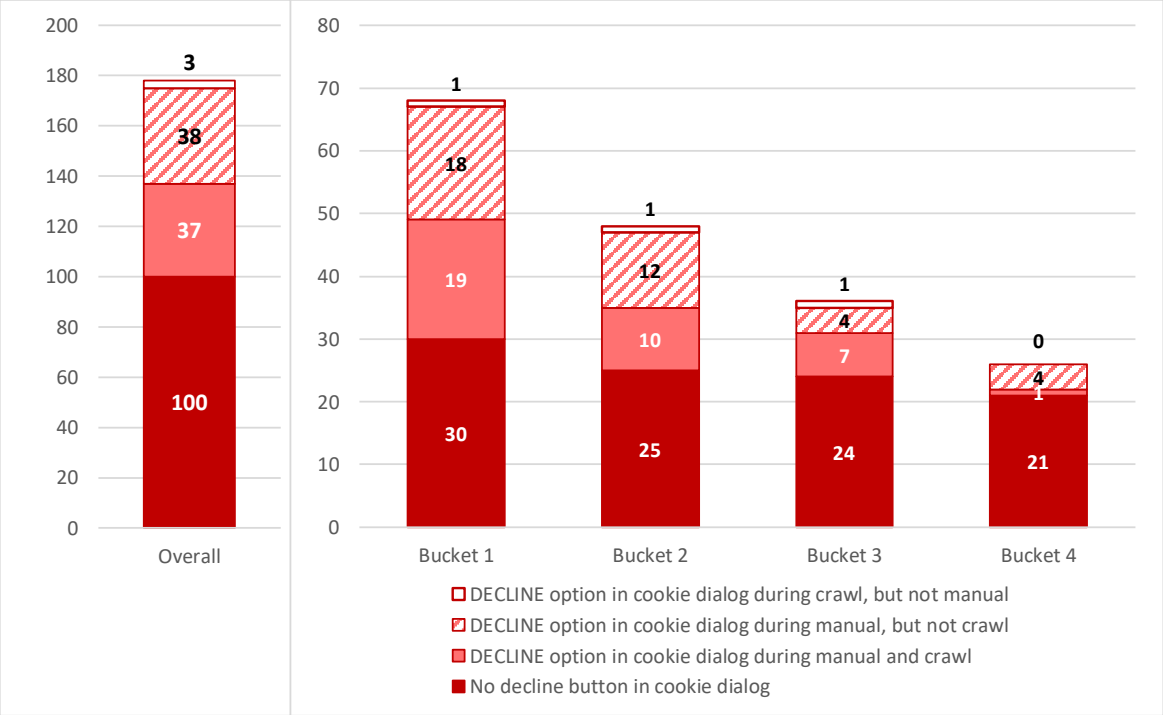


Figure 40: Detection of DECLINE option during manual visit and in the crawling session

Differences in cookies set

Figure 41 shows the differences in number of cookies set during initial visit in the manual crawl compared to the crawling session, after interaction with ACCEPT option, and after interactions with DECLINE option. A positive number means more cookies were detected during manual visit. Only websites are taken into account where both visits were successful.

Over all the buckets in the initial visit the average and median difference in cookies being set is usually very small (median = 0; average = 0.9). There are some outliers though where 55 more cookies were saved during manual visit (88 vs 33) but also where 44 more cookies were saved during the crawl session (29 vs 73). The reason for these differences in both directions is not very clear. It could be a difference in the timing of the visit, the website could set different cookies on a different day, it could also be because the automated crawler was detected, or it could also be the difference in time of the visit, some websites load more cookies the longer a visit lasts. On average though the results seem to be comparable and reliable.

Over all the buckets after interaction with an ACCEPT option the average difference in cookies being set is a little larger than during the initial visit, but also averages to around zero (median = 0; average = -0.9). The biggest outlier for the manual visits had 63 more cookies during manual visit (92 vs 29) and the biggest outlier for the crawler had 108 more cookies set during the crawl session (40 vs 148). On average though the results seem to be comparable and reliable.

The data after interaction with a DECLINE option is limited here, only 116 DECLINE options were detected during manual visit and during the crawling session. Over all the buckets the average is a gain a little larger but the median is still 0 (median = 0; average = -2.9). The biggest outlier for the manual visits had 12 more cookies during manual visit (32 vs 20) and the biggest outlier for the crawler had 56 more cookies set (48 vs 104). On average though the results seem to be somewhat reliable.

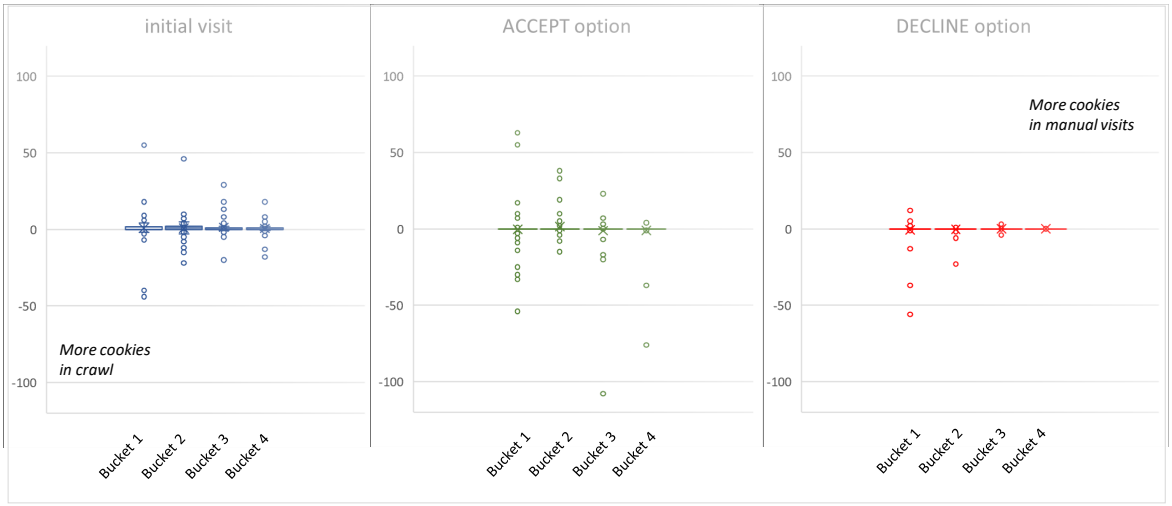


Figure 41: Difference in cookies set during initial visit in the manual visit compared to the crawling session, after interaction with an ACCEPT option and after interaction with DECLINE option

5.4.8 COMPARISON TO [AER21]

The following is a comparison of the results to the paper from Koen Aerts [Aer21] where this paper is a succession of. In the paper an older version of the Tranco list is filtered for the top 500 websites using the ccTLD from each European country. In this paper as well, the top 500 websites from each country were added to the list of website to visit using the crawler.

In Figure 42 the same six countries are re-evaluated using the same parameters. The only difference in this crawl is there hasn't been made a distinction in the type of cookie dialog that is not an iframe. Overall there is on average a reduction in cookie dialogs shown and the technology used by the cookie dialogs are less used as iframe. Overall no big differences were found.

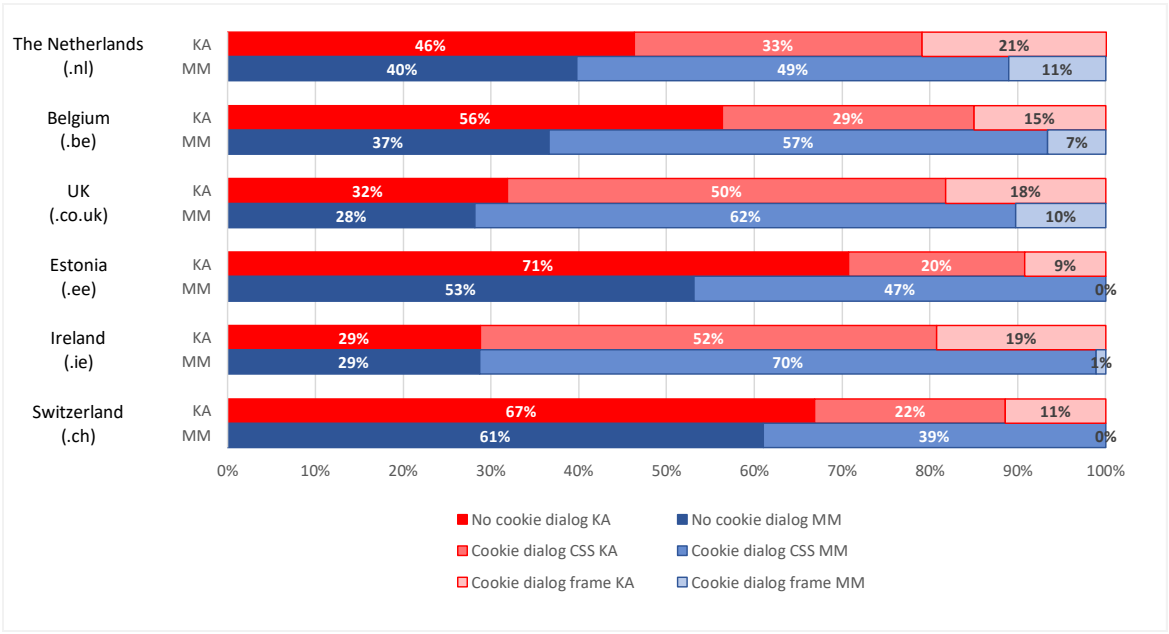


Figure 42: Comparison of cookie dialog occurrence and its technologies between Koen Aerts and Maarten Meyns

Comparison in the detection of ACCEPT option

Figure 43 shows the amount of websites from each European country an ACCEPT option was detected compared between the study done by Koen Aerts and the crawler in this study (Maarten Meyns). Results were ranked by the detection of ACCEPT option in this paper. Overall more ACCEPT options were detected during this crawl run. Some countries show big rises in detection, sometimes over three times as much ACCEPT options were detected. Unfortunately no data was found on how much cookie dialogs were found on all European countries, the results here are thus only absolute number without calculating the detection of a cookie dialog. More detailed results around this can be found in Figure 23. A conclusion however can be made here that some countries show a big rise in showing ACCEPT options compared to the previous results.

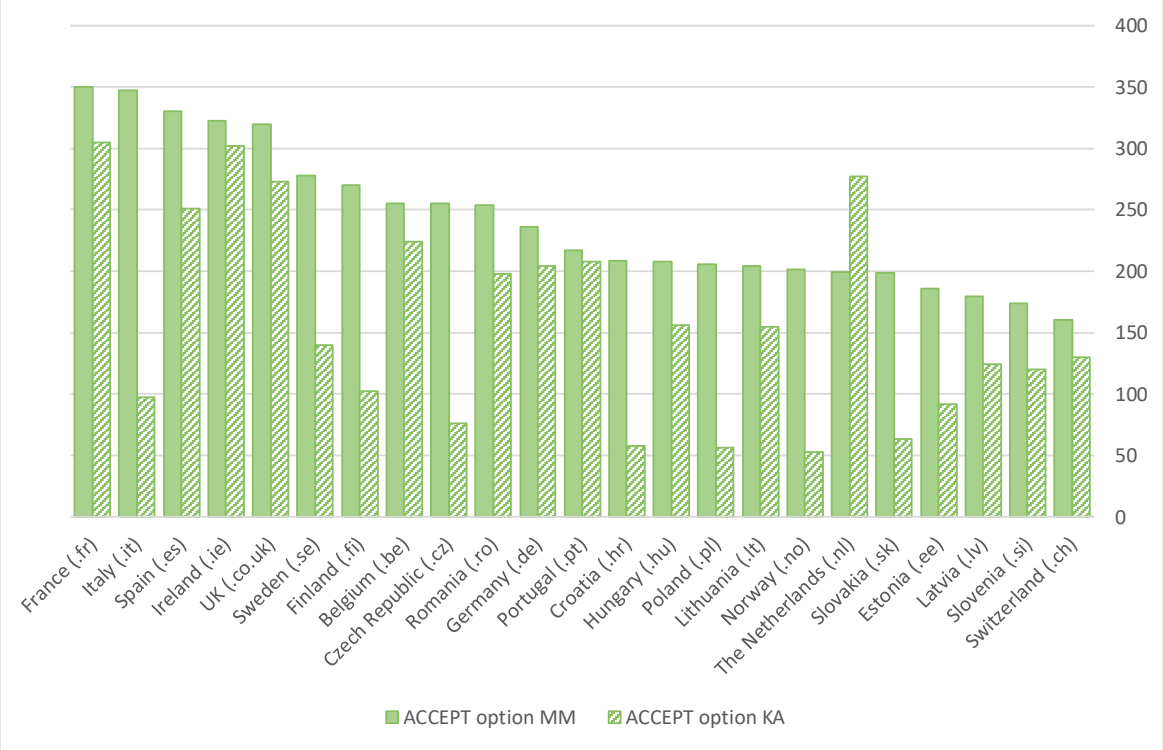


Figure 43: Comparison of the detection of an ACCEPT option between Koen Aerts and Maarten Meyns

Comparison in the detection of DECLINE option

Figure 44 shows the amount of websites from each European country a DECLINE option was detected compared between the study done by Koen Aerts and the crawler in this study (Maarten Meyns). Overall in almost all countries there is a big rise in detection of DECLINE options in this study. Previously in a lot of countries no DECLINE option was detected at all and more DECLINE options were detected now. Difference in detection between the crawlers could be part of the fluctuations, but the larger rise in DECLINE options show that websites have started implementing the DECLINE option more. The direction is clear, more and more websites have started implementing DECLINE options but there is still a long way to go.

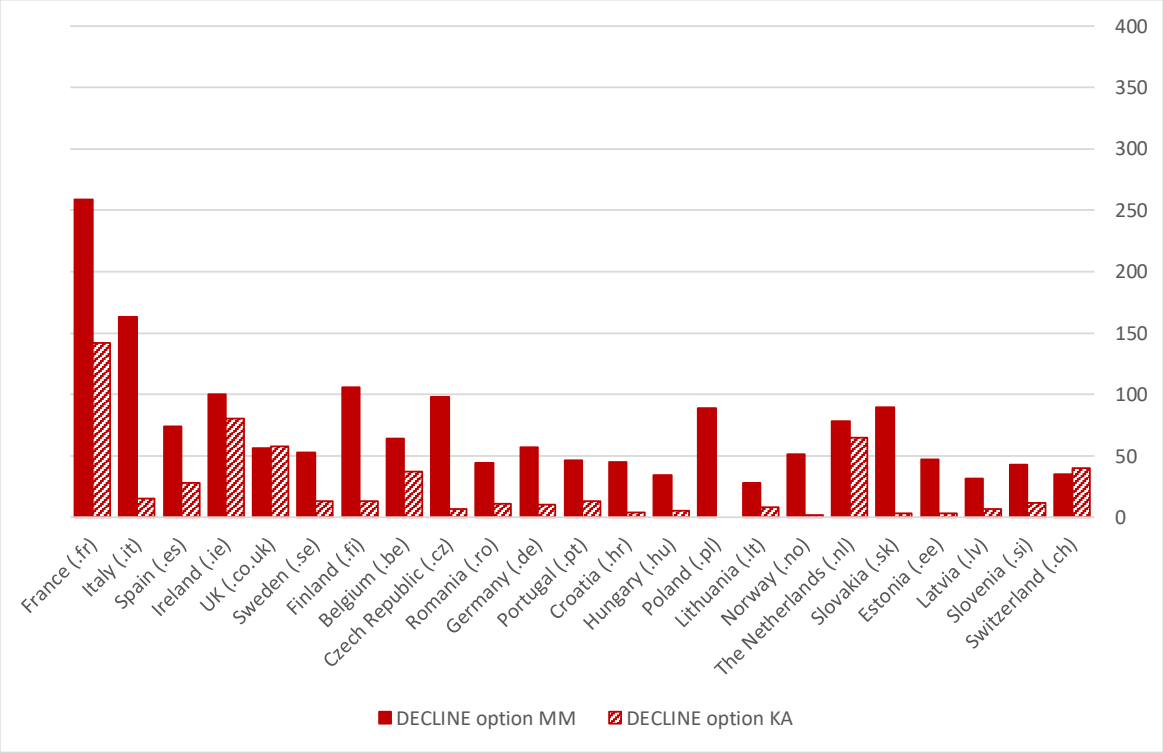


Figure 44: Comparison of the detection of a DECLINE option between Koen Aerts and Maarten Meyns

5.4.9 NO JAVASCRIPT

Although JavaScript is used to display normal website functionality, JavaScript is also used to load in advertisements, inject cookies, and to display a cookie dialog. Disabling JavaScript is one of the ways to reduce advertisements and cookies. By crawling the same websites without JavaScript enabled it will be determined if less cookies are used, and also if websites still implement a cookie dialog. If no cookie dialog is presented then no consent can be given and less cookies should be set. Although a good deal of websites will only show a landing page noticing the user the website does work without JavaScript the results can still give an insight into the usage of cookies or a cookie dialog when JavaScript is disabled.

For these results the same 100,000 websites have been revisited with JavaScript disabled. And for all graphs in this chapter the same division in buckets are used. Although there were different numbers of successfully visited websites in each bucket, the overall results stay the same.

Figure 45 shows the detection of cookie dialogs in each bucket in the no JavaScript crawling session and the normal crawling session with JavaScript enabled. The graph shows that almost all websites do not implement an alternative cookie dialog when JavaScript is disabled. All buckets show comparable numbers, independent to whether they did or did not show a cookie dialog with JavaScript enabled. Clearly most websites do not have a backup system to display a cookie dialog if JavaScript is disabled. Thus these websites cannot ask for permission to use cookies. If no permission can be given then it would be expected that less cookies are used by those websites.

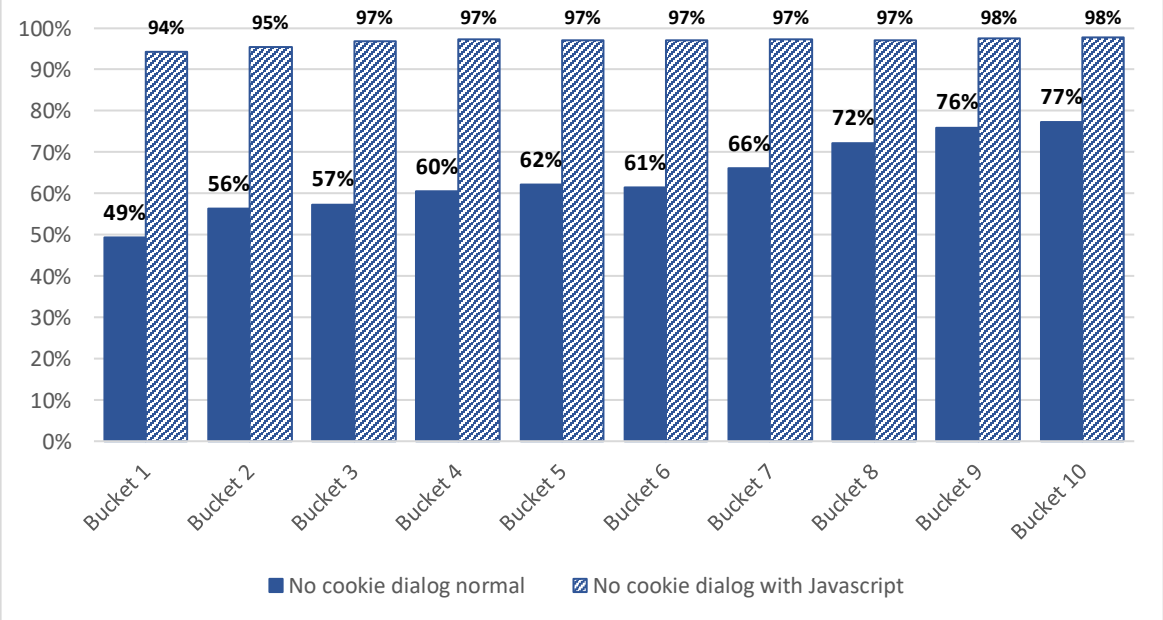


Figure 45: Percentage of cookie dialogs detected in each bucket with Javascript enabled and without Javascript

Figure 46 shows for each bucket the average first and third party cookies that are set by all websites. Compared to the crawling session (see Figure 31 a lot less cookies are set here. Although some websites were not functional without cookies there is still a large reduction in the amount of cookies that are set. Fortunately the reduction in cookie dialogs also resulted in the reduction of first and third party cookies. If the same number of cookies would have been used then that would indicate websites do not care for asking consent and still use the cookies. The next graph will determine if the reduction of cookies is linked to the displaying of a cookie dialog.

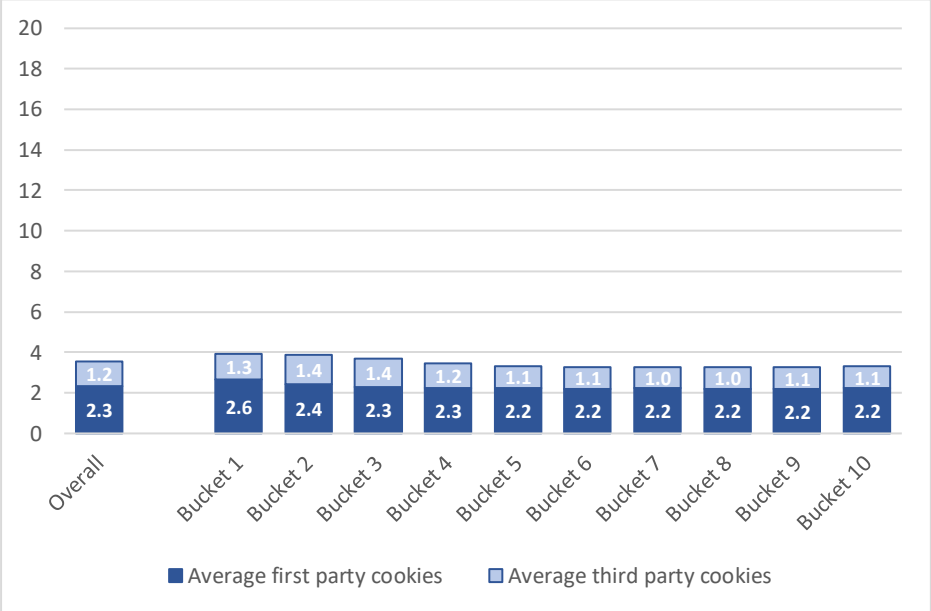


Figure 46: Average first and third party cookies set for each bucket without Javascript

Figure 47 shows for each bucket the average cookies set before interaction with a detected cookie dialog, without a cookie dialog detected, and after interacting with the ACCEPT option. Here as well compared to the crawling session with JavaScript enabled (see Figure 26 and Figure 27) there is also a large reduction in cookies set with or without a cookie dialog. And even when consent is given less cookies are used. A reduction in all of these cases suggests cookies are mainly loaded using JavaScript through either advertisements or normal website behavior.

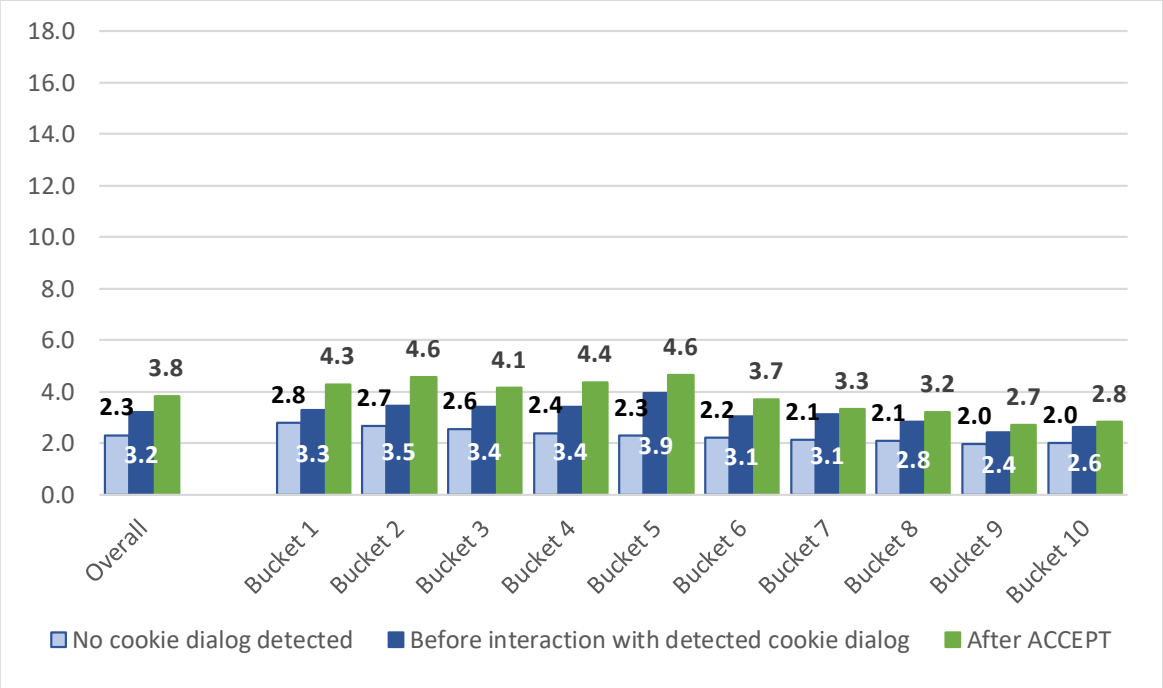


Figure 47: Average cookies set for each bucket without Javascript before interacting with a detected cookie dialog and without a cookie dialog detected

5.5 Conclusion

Although the crawler did not work flawlessly, it enabled an analysis on a representative portion of the current internet landscape. In the results the detection of cookie dialogs and the cookies that are set before or after interaction with one of the options have been analyzed.

For European countries, the detection of cookie dialogs and the ACCEPT and DECLINE option show large differences between all countries. Most interesting is the detection of the DECLINE option. This option was not detected on a large amount of websites for all countries, only France showed a decent amount of DECLINE options due to stricter applied rules. This shows that rules need to be applied stricter to have an easy to access DECLINE option in most countries.

Dividing the websites into more and less visited websites showed that more visited websites tend to show more cookie dialogs and DECLINE options. This shows that more visited websites are following regulations better or are forced to follow them better because they are more looked at.

The analysis of the cookies being set shows that on average more cookies are set when a cookie dialog is used than if no cookie dialog is used. As expected, on average more cookies are set after giving consent to using cookies then if the use of cookies is denied. Surprisingly some websites set less cookies after being allowed to use cookies, similarly some websites set more cookies after disallowing the use of cookies. An unexpected and unexplained process.

Analyzing the providers of the cookies show that some third party cookie providers were present on a staggering amount of websites, meaning they could be able to link unique visitors from multiple different websites. The presence of big providers before consent and without a cookie dialog was also very large, leaving the question if these cookies are legitimately used.

Analyzing HTTPS redirects, showed a large portion of the websites are redirecting visits to HTTPS websites for security. The proportion of the internet that does so has risen a lot in the last years. Security settings in cookies could however be implemented more, especially by smaller providers or from the websites itself.

Compared to Section 4 showed the crawler worked as expected, but detection of the cookie dialog and its options could be improved upon. There were some differences in the number of cookies being set, but those can be normal irregularities and do not invalidate the results from the crawler.

Compared to the paper from Koen Aerts [Aer21] from 2021 of which this a succession of showed that there were only small differences in the detection of the cookie dialog (also iframes usage was significantly reduced) and ACCEPT options, but rather large differences in the presence of a DECLINE option. This suggests that in the last year more websites have started implementing a DECLINE option in their cookie dialog.

At the end the effect of JavaScript has been researched. A very large portion of the websites rely on JavaScript to show a cookie dialog, have no backups technology in place and thus do not show a cookie dialog if JavaScript is disabled. A big reduction in cookies set has also been noticed with JavaScript disabled with or without a cookie dialog present and even after interacting with the ACCEPT option. This suggests most cookies are in-

jected using JavaScript either by advertisements or normal website behavior.

5.6 Discussion

It looks like the cookie dialog landscape is rapidly changing in a positive direction. Compared to last year or even during the development of the crawler some websites have changed their cookie dialog to implement clearer options for the user. This suggests regulations are working and fines do incentivize websites to be more clear in the options they serve. However not only big European countries and more visited websites need to follow regulations, all websites do need to follow them as well.

Big third party providers are using cookies on a huge proportion of the websites, with or without user permission. And although this substudy did not classify the cookies that are being set, it is clear that providers need to be regulated more. There is a good chance these providers are not following regulations. Not only do the websites have to be scrutinized in the use of cookies, cookie dialogs and its options, providers need to be looked at as well. If a provider is allowed to set an advertisement by the website for example, the provider also has to follow the choice of the user in the cookie dialog and if a cookie dialog is not available the provider should not be allowed to set cookies.

HTTPS security usage is improving but should be standard on all websites, as is the security settings in a cookie. Browsers should force HTTPS encryption on all websites and should block insecure cookies.

Third party cookies are almost only used to track users, there is almost no non-tracking cookie behavior that cannot be executed by first party cookies. Some browsers have started blocking third party cookies, but the biggest browser Chrome is continuously postponing this implementation. Regulators should make blocking third party cookies mandatory by all browsers.

Disabling JavaScript is a easy way to limit the presence of cookie dialogs and to limit the injection of cookies. Disabling JavaScript however results in some websites not working at all or some parts of the website to not work anymore. So this is not a workable solution for those websites. A working extension that selectively blocks scripts could be a valid solution. Simply automatically selecting the DECLINE option (expanded to the second screen in the cookie dialog as well) does apparently not block the same amount of cookies. And even if no cookie dialog is available there are still cookies present. Be it tracking or not. The effect of ad blocking extensions have not been studied, but as these extension block scripts from advertisers, it is expected cookies are also reduced when using ad blocking extensions.

5.7 Future work

It was initially foreseen to use the data generated by the crawler concerning the design of the cookie dialog and its options to classify all websites into the use of dark patterns. This research will be completed in a different paper. If the date this new paper is produced on is at a later date than this paper then the crawl could be reran to get a comparison between older data and newer data. To see if websites are changing their implementations and to get more recent data.

Not all cookie dialogs and not all options were detected by the crawler. There needs to be more effort put into preparing the cookie dialog candidates as well as the options in a cookie dialog. Extra effort needs to be put into the machine learning models. Although the cookie dialog prediction has a high success rate, for the options improvement is needed. Work also needs to be done to be able to use the MODIFY option and interact with all options in the next dialog to deny the usage of all cookies. The SAVE option was also used too little and was too unreliable to have a meaningful contribution.

Cookies collected during the crawling session have not been classified. Section 4 discusses the options to classify cookies and there is currently no option available to automatically classify all the cookies in a fast and reliable way. Although `Cookiepedia.co.uk` has a very large database of cookies, this database is not easily accessible for the end user. Future work will need an accessible database if it would want to classify all cookies. The analysis of the details of the cookies being set can also be expanded on. The expiry date or the value from the cookies could also be studied.

REFERENCES

[Aer21] Koen Aerts. Cookie dialogs and their compliance. July 2021. 9, 35, 38, 44, 70, 76

[BKCB22] Dino Bollinger, Karel Kubicek, Carlos Cotrini, and David Basin. Automating Cookie Consent and GDPR Violation Detection. In *31st USENIX Security Symposium*. USENIX, 2022. Prepublication. 10

[EEZ+15] Steven Englehardt, Chris Eubank, Peter Zimmerman, Dillon Reisman, and Arvind Narayanan. OpenWPM: An automated platform for web privacy measurement. March 2015. Draft. 38

[EN16] 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*, pages 1388–1401. ACM, October 2016. 35, 63

[ERE+15] 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*, pages 289–299. IW3C2, May 2015. 7

[GKB+18] Colin M. Gray, Yubo Kou, Bryan Battles, Joseph Hoggatt, and Austin L. Toombs. The Dark (Patterns) Side of UX Design. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–14. ACM, April 2018. 8

[HvdSB19] Chris Jay Hoofnagle, Bart van der Sloot, and Frederik Zuiderveen Borgesius. The European Union general data protection regulation: what it is and what it means. *Information & Communications Technology Law*, 28(1):65–98, January 2019. 8

[JKV19] Hugo Jonker, Benjamin Krumnow, and Gabry Vlot. Fingerprint Surface-Based Detection of Web Bot Detectors. In *Computer Security – ESORICS 2019*, volume 11736, pages 586–605. Springer, 2019. 9, 39, 41, 94

[KNHF22] Rishabh Khandelwal, Asmit Nayak, Hamza Harkous, and Kassem Fawaz. CookieEnforcer: Automated Cookie Notice Analysis and Enforcement. In *arXiv:2204.04221 [cs]*. arXiv, April 2022. 10, 38, 40, 41

[Kri01] David M. Kristol. HTTP Cookies: Standards, privacy, and politics. In *Transactions on Internet Technology*, volume 1, pages 151–198. ACM, November 2001. 7

[LPVGT+19] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczynski, and Wouter Joosen. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings 2019 Network and Distributed System Security Symposium*. Internet Society, 2019. 9

[MAF+19] Arunesh Mathur, Gunes Acar, Michael J. Friedman, Elena Lucherini, Jonathan Mayer, Marshini Chetty, and Arvind Narayanan. Dark Patterns at Scale: Findings from a Crawl of 11K Shopping Websites. In *Proceedings of the ACM on Human-Computer Interaction*, volume 3, pages 1–32. ACM, November 2019. 8

[MBS20] 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:1911.09964 [cs]*. arXiv, February 2020. 9, 36

[RGBZ16] Oliver Radley-Gardner, Hugh Beale, and Reinhard Zimmermann. *Fundamental Texts On European Private Law*. Hart Publishing, 2016. 8

[SRDK+19] 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*, pages 340–351. ACM, July 2019. 9

A COOKIE DIALOG EXAMPLES

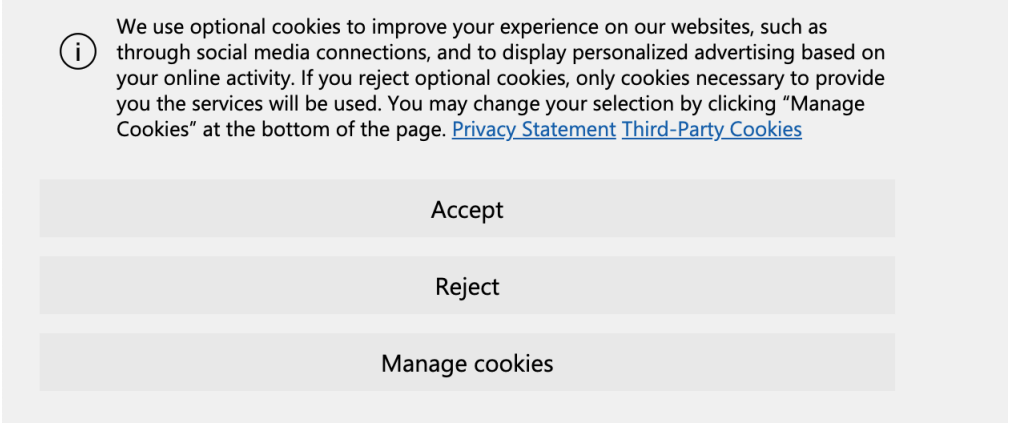


Figure 48: Possible actions microsoft.com

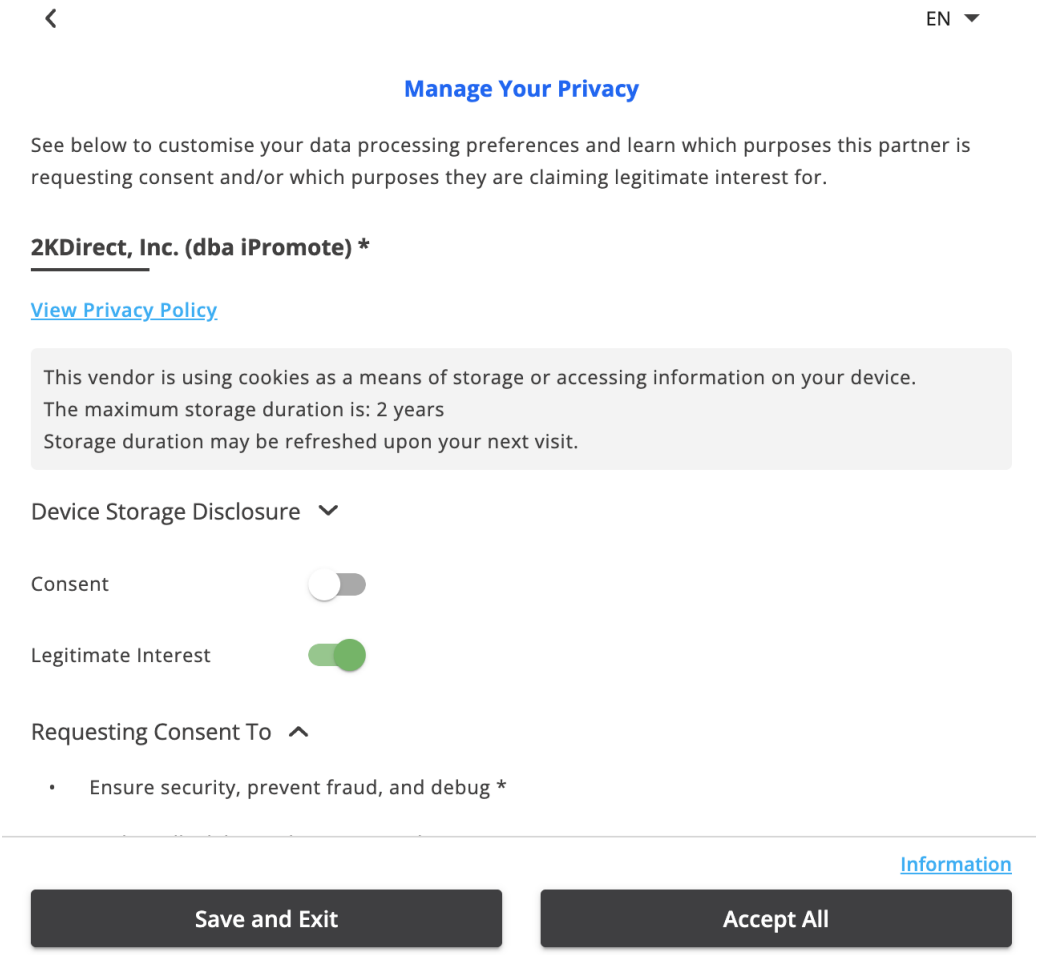


Figure 49: Possible actions zeelandnet.nl

B DOMAIN SELECTION

There are 400 selected domains divided over four buckets according to the bucket composition scheme from table 3.

BUCKET 1: 1-5,000

7; microsoft.com
13; linkedin.com
14; apple.com
26; office.com
51; wordpress.com
67; intuit.com
101; wellsfargo.com
113; medium.com
138; bbc.com
154; etsy.com
182; issuu.com
199; reuters.com
230; blogger.com
243; shopify.com
248; alibaba.com
272; ilovepdf.com
284; udemy.com
286; espn.com
294; steampowered.com
305; doi.org
311; unsplash.com
355; steamcommunity.com
356; snapchat.com
363; marriott.com
369; digitalocean.com
374; huawei.com
376; mailchimp.com
411; mysql.com
457; theverge.com
467; stackexchange.com
499; zoho.com
505; bankofamerica.com
510; nbcnews.com
594; nike.com
657; capitalone.com
675; redhat.com
751; gofundme.com
806; getpocket.com
853; jquery.com
857; smallpdf.com
936; timeanddate.com
954; zerodha.com
981; getbootstrap.com
1010; nintendo.com
1012; substack.com
1214; google.nl
1225; discogs.com
1239; fool.com
1256; redbubble.com
1316; citrix.com
1409; indianexpress.com
1412; instructables.com
1425; venturebeat.com
1444; tutorialspoint.com
1448; northwestern.edu
1518; enable-javascript.com
1524; fireeye.com
1573; ahrefs.com
1625; auth0.com
1628; moz.com
1667; thawte.com
1702; boston.com
1822; myfitnesspal.com
1831; pruffme.com
1918; lonelyplanet.com
2248; blackberry.com
2258; zapier.com
2284; zerohedge.com
2298; bustle.com
2502; ti.com
2503; calameo.com
2530; unpkg.com
2670; ipcc.ch
2680; zynga.com
2765; eu.com
2766; synology.com
2832; pypi.org
2872; curl.se
2897; pngtree.com
2990; pixlr.com
2991; freelancer.com
3079; livestrong.com
3118; perl.com

3128; nairaland.com
3140; oregonlive.com
3236; metlife.com
3256; marthastewart.com
3278; gtmetrix.com
3429; telenet.be
3508; sbnation.com
3593; vodafone.com
3827; producthunt.com
4253; cc.com
4319; discovermagazine.com
4320; hwg.org
4366; xs4all.nl
4498; shell.com
4650; foodandwine.com
4675; virginmedia.com
4700; imgbb.com

BUCKET 2: 5,001-25,000

5363; rwth-aachen.de
5380; mindbodygreen.com
5467; instyle.com
5489; qvc.com
5663; owasp.org
5689; townhall.com
5800; sparknotes.com
5862; questdiagnostics.com
6052; thediplomat.com
6053; vrt.be
6161; invisionapp.com
6331; traveloka.com
6379; eatingwell.com
6563; shape.com
6601; optum.com
6634; nngroup.com
6674; pillpack.com
6837; petapixel.com
6882; solarwinds.com
6956; slashfilm.com
7017; opencart.com
7097; wonderhowto.com
7254; rijksoverheid.nl
7309; reolink.com
7316; shopstyle.com
7504; actblue.com
7613; vista.com
7789; villagevoice.com
7823; symbolab.com
7836; pch.net
8051; goodmorningamerica.com
8159; torrentfreak.com
8457; hightail.com
8510; mikrotik.com
8519; thechive.com
8635; audioboom.com
9031; bepress.com
9094; listverse.com
9154; omnicalculator.com
9333; bravotv.com
9403; nrc.nl
9446; gitbook.com
9605; annualcreditreport.com
9824; restream.io
10448; rabobank.nl
10478; celebritynetworth.com
10607; tcpshield.com
10622; mondaq.com
10969; viagogo.com
11158; wp-royal.com
11272; rapidapi.com
11339; rubyonrails.org
11386; octafx.com
11462; sixt.com
11579; onlineradiobox.com
11692; scienceblogs.com
12231; yubico.com
12351; steinberg.net
12615; tldp.org
13160; voxmedia.com
13257; vlaanderen.be
13346; toyota-europe.com
13425; virtualmin.com
13432; thebump.com
14119; realself.com
14206; nabble.com
14265; postnl.nl
14351; pixlee.com
14457; zattoo.com
14677; typingtest.com
14959; shinyapps.io
15017; ziggo.nl
15023; sce.com
15694; rubygems.org

15732; vestiairecollective.com
16638; proximus.be
16779; sass-lang.com
16932; terraform.io
17014; webassign.net
17564; usra.edu
17566; zf.com
18135; thecinemaholic.com
18775; reviewed.com
18918; osuosl.org
19327; sonoma.edu
19921; nyulangone.org
21512; oodle.com
21537; scrum.org
21895; unknowncheats.me
21902; willyweather.com
21992; zipcar.com
22386; smartprix.com
22548; romsfun.com
22580; teachertube.com
23223; oxygen.com
23840; taschen.com
24061; rankfirsthosting.net
24459; secure-is.nl
24586; tec-it.com
24690; versobooks.com

BUCKET 3: 25,001-100,000

25051; mailtrap.io
25966; rvo.nl
26077; sporza.be
26521; vodafone.nl
26812; lifestyleasia.com
27437; westmancom.com
28192; shockmedia.nl
29577; nomeo.be
29922; nlnetlabs.nl
30934; wallpaperscraft.com
32082; zeelandnet.nl
32121; urlebird.com
32474; wsgr.com
33237; snsbank.nl
34593; sarthaks.com
34690; readwritethink.org
35330; themortgagereports.com
35477; writing.com
35678; upload.ee
35734; techterms.com
36384; santafe.edu
37033; wma.net
37043; neliti.com
38086; powerdns.com
38579; mediamarkt.nl
38923; radaris.com
38954; menuism.com
39208; spineuniverse.com
41295; wanmacxe.com
41346; mysqltutorial.org
41368; posteo.de
42049; telemetr.io
42877; virtamate.com
43160; zetcode.com
44290; wanikani.com
44850; siemens-energy.com
45175; libsdl.org
45802; luckymodapk.com
46608; setcialimir.com
47334; nakedwines.com
48243; mudasure.com
48594; lotro.com
48957; wtguru.com
50813; oneplace.com
51115; nederlandwereldwijd.nl
51191; tips.net
51195; publicintelligence.net
51526; tindie.com
52392; videocardbenchmark.net
53437; wpsoul.com
53552; netflixtechblog.com
54522; mibbit.com
55481; practicalpainmanagement.com
55947; plusportals.com
56076; seegore.com
56280; rocketcyber.com
56289; seotoolbuy.com
57012; ogc.org
57343; techbout.com
57786; tripadvisor.be
57947; paydayloansgeorgia.net
58496; mailfence.com
58742; transcy.io
58963; wan-ifra.org
59038; positronx.io

59226; mijnwoordenboek.nl
59255; thorlabs.com
60274; rugs.com
60629; veraxen.com
60650; letssingit.com
60685; rtrt.me
62217; thefarside.com
62608; moviemeter.nl
63371; readwritenews.com
63858; macromates.com
63931; time.nl
64252; public-sourcing.be
65643; wrestlingheadlines.com
66265; permies.com
66433; prana.com
66445; zid.com
66531; pri-li-gy.com
67469; lumc.nl
67763; rgs.org
67858; myopenmath.com
68077; lightningbase.com
68119; your-webhost.nl
68136; wolfandbadger.com
68565; madsci.org
69887; uzbrussel.be
70601; purposegames.com
70627; onlinepngtools.com
70728; ringba.com
71214; zonnet.nl
72650; techandtrends.com
73259; triphouserotterdam.nl
73503; webwinkelkeur.nl
73828; nftcalendar.io
74326; savingcountrymusic.com
74505; magicspoiler.com

BUCKET 4: 100,001-1,000,000

107256; twinery.org
110913; virgingames.com
117026; modpodgerocksblog.com
124588; trackbill.com
125603; readopm.com
127864; researchmaniacs.com
131660; starbike.com
144935; polyroche.com
146962; thebulletin.be
151524; recruitmentresult.com
152202; prijs-parfum.nl
154669; shambhala.org
162554; provalue.nl
170978; minimum-wage.org
173310; tradechaser.com
188123; sildenafilnoprescription.com
189685; motorsportsales.com
190288; skiomusic.com
191391; twagoda.com
193617; openjdk.org
201142; styledemocracy.com
218395; onepeterfive.com
221256; ricaud.com
223150; nufc.com
223470; plutodesk.com
228787; procor.be
241510; quasardata.com
259060; stadiumgaming.gg
260684; nashvillesmls.com
260686; lctraumacoalition.org
274411; ubitennis.net
280527; xolairhcp.com
285304; ledel-europe.com
286319; skepticalraptor.com
289806; microschools.com
292191; viagravpills.com
295950; viewacr.com
298018; petefreitag.com
300419; webtic.nl
312891; serversforhackers.com
335064; searchads.com
348277; newsreel.org
353569; websolid.be
363625; trans-video.net
374572; rottzgames.com
383276; lordswm.com
389740; mangabob.com
398495; socialarchitects.nl
400501; singlepanda.com
410844; nxthost.com
413135; lvhtebook.com
421802; sentencechecker.org
439470; nicetightash.com
448583; pioneers.io
463467; moviesclue.com
471964; rijksvastgoedbedrijf.nl

472160; nlt-media.com
494768; krakenrum.com
501615; trackdrive.net
533212; thetreemaker.com
542579; qualityinfo.org
544477; technomadia.com
547441; oldgameshelf.com
548678; petrockblock.com
553409; online-cpp.com
561832; peakplex.com
569563; relume.io
569784; stoprugpull.com
571478; realmanage.com
589349; mfwbooks.com
596152; vtk.be
604583; vatcalconline.com
604674; thetechoutlook.com
609873; writemonkey.com
617953; spire.net
632193; wxblocker.com
646667; vpnnext.com
657325; tradersdiaries.com
657695; trinityacademy.org
665929; thegoaspotlight.com
667330; playemulator.com
675702; wasatch.com
705740; mmotank.com
708770; thehandprints.com
720925; pbtresourceline.com
731075; prolife.nl
764763; libmir.com
775047; mgwater.com
795643; linkdirectory.be
796623; luxuryactivist.com
800871; vistaweb.nl
811650; qfimr.com
815490; majorcitieschiefs.com
818985; maxpay.com
825194; sab.org
831196; zezam.io
852520; methocarbamolrobaxin.com
857018; nidi.nl
868046; simplywebservices.net
916741; vaarkaartnederland.nl

C SQLITE DATABASE

SQLite database

The database consists of four tables. *elements* containing information from the elements in the website and if they are visited, *cookies* containing the information from the cookies, *predictions* containing all predictions made and *visits* containing all the visits that are made. Figure 50 shows the tables and the columns of it.

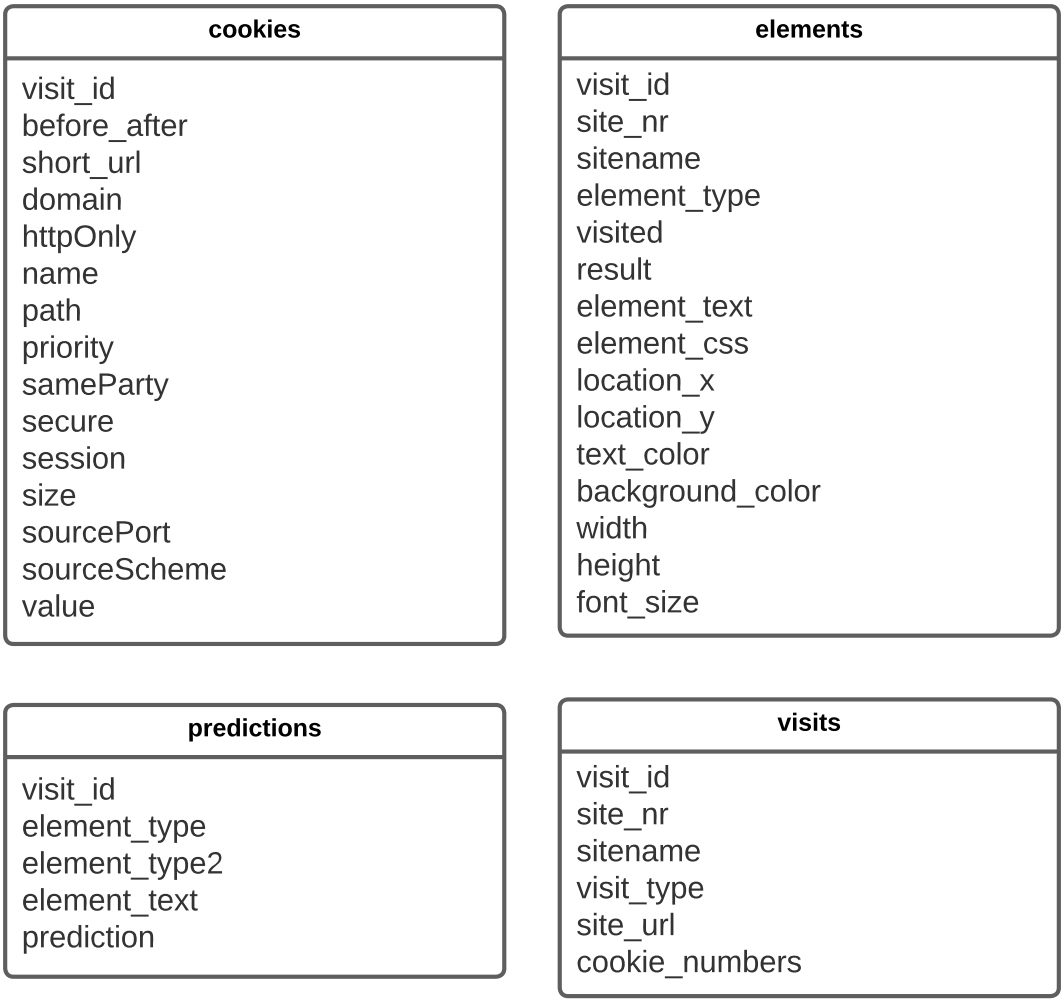


Figure 50: Database structure

The *elements* table is the most important part of the database. It contains the websites to be visited, and if it has been visited. Each website has the element type 0 meaning this is an initial visit. During this initial visit the cookie dialog will be detected if present, and the row will be populated with information from that visit: The visit id that is used to link to other tables, the result of the visit and the design information of the cookie dialog if found. If options from the cookie dialog were detected these will be saved as new rows and flagged as not visited.

The *cookies* table contains the details of all cookies. Using the visit-id and element-type

this information can be linked back to the *elements* table. The essential information of each individual cookie is saved in each row. Below is a non extensive list of the information from the cookie that is saved.

- Domain URL (a third party cookie will show a different domain URL)
- Time until expiry (-1 meaning when session ends)
- Name of the cookie
- Value stored in the cookie
- Security flags

The *predictions* table contains all predictions that were made. Using the same visit-id this information can be linked back to the *elements* table. The prediction category, the HTML element-type, the element-text and the prediction is saved. This data can be used to validate each prediction.

The *visits* table contains a list of all visits made by the crawler containing the visit-id and the actual URL of the visit. Not all visits are saved in all tables, and rows can be overwritten in other tables if a website is revisited. This list of unique visits can be used to determine which websites were revisited and enables the analysis of HTTPS redirects.

D MODULES USED FOR CRAWLING

Below is a brief summarizing of the tools used to form the crawling artifact.

Python
Python has been chosen as the coding language. Although it is a relatively new language and is not used during the education it is chosen because it is a very easy to learn language and is an already widely used and documented language. There are also a lot of libraries available that can easily be integrated into this project.

PyCharm[41]
The Python artifacts have been written in the coding environment PyCharm. PyCharm is an easy to use coding software that helps setting up a coding environment and helps the user during coding, writing working code and reducing errors.

Multiprocessing[42]
Multiprocessing is used for running multiple thread at the same time. This library is used to set up the threads and also maintain persistent and atomic variables between different threads.

Sqlite3[43]
Sqlite3 is used to connect to an SQLite database. In this SQLite database all information from the crawl session will be saved. This library can execute queries to read or write to a database file.

Selenium[44]
Selenium is used to simulate a browser session. This library uses the Chrome webdriver[45] to spawn a browser session and visit a website. The webdriver can get elements from a webpage, click on an element, get the element information and get the cookies from a browser session. Browser side scripts can also be executed.

Simpletransformers[46]
Simpletransformers is used to set up a machine learning model used for the classification of the cookie dialog and the classification of the options in the cookie dialog. This library can train a classification model and use the model to make predictions.

SimplePyGUI[47]
SimplPyGUI is used to set up a simple GUI. Although everything runs in the console, this is used to display the resources that are in use: RAM, CPU and threads. It is also used to pause and gracefully stop the crawl session if needed and to automatically adjust the number of threads given the resources available.

[41] https://www.jetbrains.com/pycharm/
[42] https://docs.python.org/3/library/multiprocessing.html
[43] https://docs.python.org/3/library/sqlite3.html
[44] https://pypi.org/project/selenium/
[45] https://chromedriver.chromium.org/
[46] https://simpletransformers.ai/
[47] https://www.pysimplegui.org/en/latest/

E MACHINE LEARNING MODELS

This is an explanation how the machine learning models were set up.

Cookie dialog model. A stripped down version of the crawling artifact is used to extract the cookie dialogs (and subsequently the option candidates) for the first 500 websites from the Tranco list. These 500 visits delivered 650 cookie candidates where the text from each candidate was saved into a list. Each item from this list was manually completed with True if it is a cookie dialog and False if it is not based on the text and a screenshot from the page and candidate. Table 13 shows a few True examples and Table 14 shows a few False examples.

Option text	Class
"*We use cookies to make interactions with our website and services easy and meaningful, and better understand how they are used and to tailor advertising. You can read more and make your cookie choices here. By continuing to use this site you are giving us your consent to do this. Learn More GOT IT*"	TRUE
"*Zoom uses cookies and similar technologies as strictly necessary to make our site work. We and our partners would also like to set additional cookies to analyze your use of our site, to personalize and enhance your visit to our site and to show you more relevant content and advertising. These will be set only if you accept. You can always review and change your cookie preferences through our cookie settings page. For more information, please read our Privacy Statement. DECLINE COOKIES ACCEPT COOKIES COOKIES SETTINGS*"	TRUE

Table 13: Example of True classification of cookie dialogs (taken from zoom.com and bitly.com)

Option text	Class
"*Why Bitly? Solutions Features Pricing Resources Log in Sign up Free Get a Quote*"	FALSE
"*Brand Your Links with a Custom Domain Custom domains replace the "bit.ly" in your short links with the name of your choosing, so you*"	FALSE

Table 14: Example of False classification of cookie dialogs (taken from bitly.com)

This list was then fed into a machine learning model using the holdout method[48]. The list is split into three groups. A training group using 70% of the candidates, an evaluation group with 15% of the candidates and a prediction group using the remaining 15%. The training group and evaluation group are used to train the model and the prediction group will determine if the model can successfully predict unlearned data.

Using a random shuffle the groups are split and fed into the model using the following settings:

- Number of training epochs: 6
- Learning rate: 2e-05

[48] `https://www.geeksforgeeks.org/introduction-of-holdout-method/`

- Evaluation an train batch size: 32 (making this higher reduces the time spent but increases the RAM used)
- Maximum sequence length: 512 (This was the highest number it could go for training the amount of text in the cookie dialog, a sliding window could also be used if the text is larger than this but this deteriorated the results)
- Encoding: utf-8
- Evaluation during training
- No multiprocessing for evaluation (using multiprocessing gave errors)
- Data was validated each epoch
- Data was adjusted to lower case (mixing lower and higher case gave different results for predictions even though the raw text didn't change)
- Use early stopping techniques when mcc doesn't improve or improvement falls below 0.001 for 5 times, evaluated every 75 steps. Mcc is a coefficient based on the evaluation data (1 meaning perfect evaluation and -1 meaning all wrong predictions)

Afterwards the model was used to make predictions of the whole dataset as well to check the validity of the model and to check if the classification dataset did not contain errors. Training the model took around 1h 42min, 1m20s for the small prediction and around 9min was needed to do a full prediction. During training between 5-10GB RAM was used and between 50-70% CPU.

- Prediction results of prediction data: 96 right – 2 wrong (= 2%)
- Prediction results of full data set: 643 right – 7 wrong (= 1.1%)

After evaluating the wrong predictions it was determined during the preparation 5 cookie dialogs were wrongly classified. This actually means the model already showed better results than a user. This resulted in the final predictions to have a 648 right predictions and 2 wrong predictions (= 0.3%).

Options model. The previously trained machine learning model was used to derive cookie dialog elements from the first 500 websites from the Tranco list. Elements from the detected cookie dialogs were extracted if they could be an option (this can be a button, a link, a span or an svg). This resulted in 1155 option candidate elements where the text from each element was saved into a list. The list was manually completed with classifications using five classes depicted in Table 15.

Options	**Declaration**
ACCEPT	Accept all cookies
DECLINE	Decline all cookies
MODIFY	Modify what cookies a user will accept or decline
SAVE	Save the settings for the selection of cookies
OTHER	Something unrelated to the above

Table 15: Options list

Table 16 is the classification of the cookie dialog found in zoom.us seen in **??**.

Option text	**Class**
Privacy Statement	OTHER
DECLINE COOKIES	DECLINE
ACCEPT COOKIES	ACCEPT
COOKIES SETTINGS	MODIFY
CONFIRM MY CHOICES	SAVE

Table 16: Classification of options on `zoom.us`

The same setup was used as the previous model. The dataset was split into 3 groups using the same settings but with these alterations. The number of training epochs was raised to 25, the maximum sequence length was reduced to 64 and the stopping technique was evaluated every 100 steps. The higher number of training epochs was needed to reach a working model, presumably because of the higher number of classes and/or number of entries in the dataset. The sequence length was reduced because of length of the text on the options was a lot smaller than the cookie dialog speeding up the training process. Training the dataset took around 1h 17min. 12s was needed to do the small predictions and 1m20s for the full prediction. The load during training was roughly the same as the previous training.

- Prediction results of prediction data: 171 right - 3 wrong (= 1.7%)
- Prediction results of full data set: 1144 right - 11 wrong (= 0.95%)

Validation. Websites 500-699 of the Tranco list were manually validated. Of those 140 were properly visited (the other websites produced an error, were unreachable or were unreadable). Those 140 websites delivered 186 potential cookie dialog candidates. All predictions for these candidates of those were correctly made: 81 contained a cookie dialog, of which 77 were detected and correctly predicted, 4 cookie dialogs were not detected. Thus the predictions are 100% correct, but on 4 websites of the 140 visited a cookie dialog remains undetected (=2.8%). **??** show a cookie dialog with an ACCEPT, DECLINE and MODIFY option.

The 77 detected and predicted cookie dialogs delivered 352 option candidates. These were the predictions:

- 72 ACCEPT (1 wrong, 71 correct)
- 30 DECLINE (30 correct)
- 51 MODIFY (51 correct)
- 4 SAVE (2 wrong, 2 could have been in another category as well)
- 195 OTHER (5 wrong, 190 correct)

Most errors were options that were classified as OTHER instead of ACCEPT or DECLINE. And due to the small occurrences of the SAVE class (in the training data) this class was

not detected correctly. Because of the nature of the sometimes ambiguous text used by websites it was even not always possible for a normal end user to determine what the text from an option would mean. ACCEPT and DECLINE were usually unambiguous but MODIFY and SAVE were not always that straightforward.

In light of this one occurrence of ACCEPT and DECLINE were not correct meaning 1 of 102 predictions were incorrect (= 1%). In one occurrence the model even detected a DECLINE option that was in a different language as the other text and thus could not have been detected by a normal user. If looking at all predictions there are 8 wrong predictions and 344 correct predictions (= 2,3%).

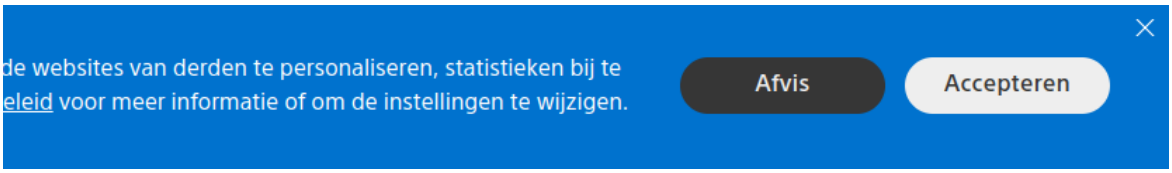


Figure 51: Wrong language in option

F SELENIUM

Setup

Windows 10 64-bit was chosen as development environment because it was the workstation of the user developing the software. Preliminary test of the artifacts are executed on the workstation. Simultaneous threads were kept low for stability, running too many threads at the same time quickly halted any operation on the workstation. The full crawl has been executed on Ubuntu 22.0.4 LTS for speed and security. Ubuntu allowed running up to 16 threads without problems, and using a fresh Ubuntu installation with no access to the Windows workstation made sure if security issues would arise the data on the workstation was not affected. Afterwards the Ubuntu installation could easily be removed. The artifacts are written in Python using PyCharm coding environment.

Selenium

Selenium has been chosen to set up the browser sessions using the Chrome webdriver. Selenium using the Chrome webdriver is the most used combination for automated visits, is well documented and very flexible. This combination and using a headful browser is also the least detected crawler according to Jonker et al. [JKV19]. Other settings mentioned were also set to avoid being detected.

Selenium alternatives

There has been a development of a few Selenium alternatives that try new methods to stay undetected. Undetected-chromedriver[49] and Selenium-stealth[50] are two alternatives. While these are successful in masking the presence of an automated browser, they do however have repercussions in the setting of cookies. Sometimes 50% less cookie were set using any of these alternatives.

Other settings

Other settings are also taken from Jonker et al. [JKV19]: Setting a common user agent without mention of Selenium or the webdriver, hiding the webdriver information in the browser file and hiding the webdriver in the navigator. Below are the settings used to start the browser session.

```
user_agents = ["Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
                537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.36",
               "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/
                537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.36",
               "Mozilla/5.0 (Windows NT 10.0) AppleWebKit/
                537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.36",
               "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
                537.36 (KHTML, like Gecko) Chrome/101.0.4951.67 Safari/537.36",
               "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
                537.36 (KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36"]

options = webdriver.ChromeOptions()
options.add_argument("user-agent=" + user_agents[visit_type])
options.add_argument("--disable-automation")
options.add_argument("--no-sandbox")
options.add_argument("--disable-dev-shm-usage")
```

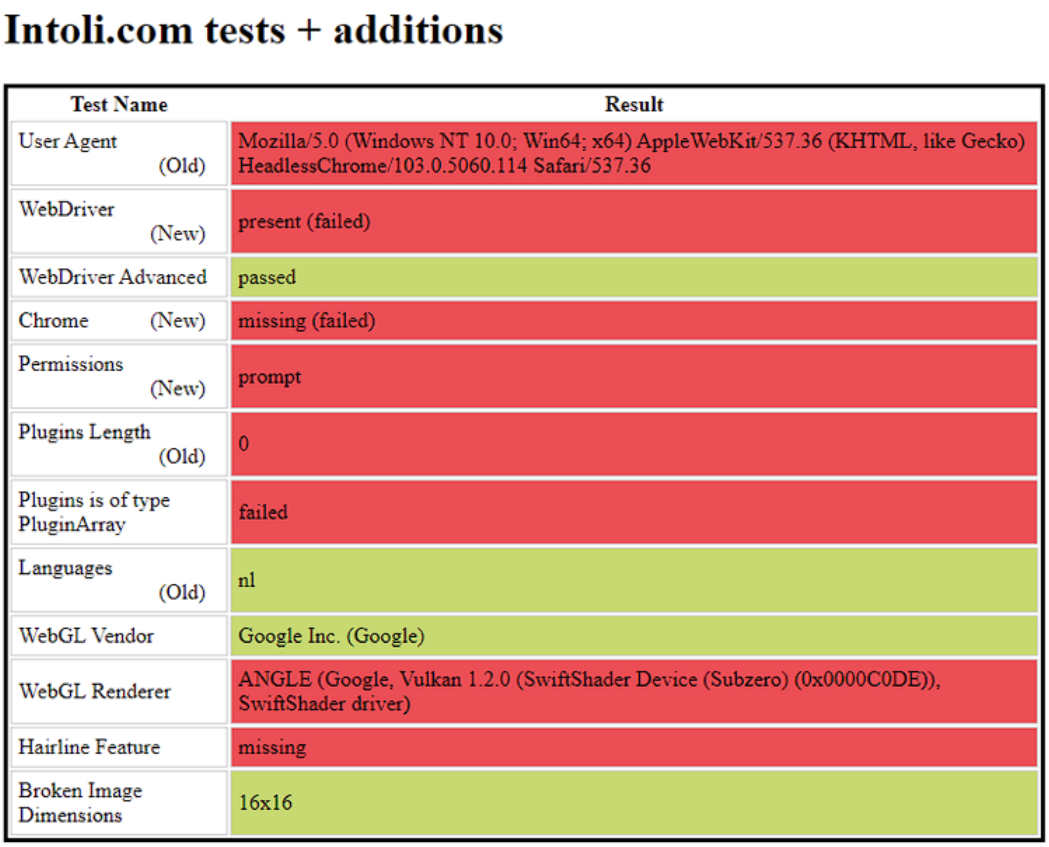
[49] https://github.com/ultrafunkamsterdam/undetected-chromedriver
[50] https://pypi.org/project/selenium-stealth/

```
options.add_argument("--disable-browser-side-navigation")
options.add_argument("--dns-prefetch-disable")
options.add_experimental_option("useAutomationExtension", False)
options.add_experimental_option("excludeSwitches", ["enable-automation"])
options.add_argument('--useAutomationExtension=false')
options.add_argument('--disable-blink-features=AutomationControlled')

driver = webdriver.Chrome(service=Service(PATH_CHROME), options=options)

driver.set_window_size(1920, 1080)
driver.execute_script("Object.defineProperty(navigator, 'webdriver',
    {get: () => undefined})")
```

There is an automated website that shows how well the webdriver is being detected[51]. Figure 52 shows the result without masking measures implemented and a headless session. Figure 53 shows the result with masking measures implemented and a headful session.



Intoli.com tests + additions

Test Name	Result
User Agent (Old)	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/103.0.5060.114 Safari/537.36
WebDriver (New)	present (failed)
WebDriver Advanced	passed
Chrome (New)	missing (failed)
Permissions (New)	prompt
Plugins Length (Old)	0
Plugins is of type PluginArray	failed
Languages (Old)	nl
WebGL Vendor	Google Inc. (Google)
WebGL Renderer	ANGLE (Google, Vulkan 1.2.0 (SwiftShader Device (Subzero) (0x0000C0DE)), SwiftShader driver)
Hairline Feature	missing
Broken Image Dimensions	16x16

Figure 52: `https://bot.sannysoft.com/` without masking

[51] `https://bot.sannysoft.com/`

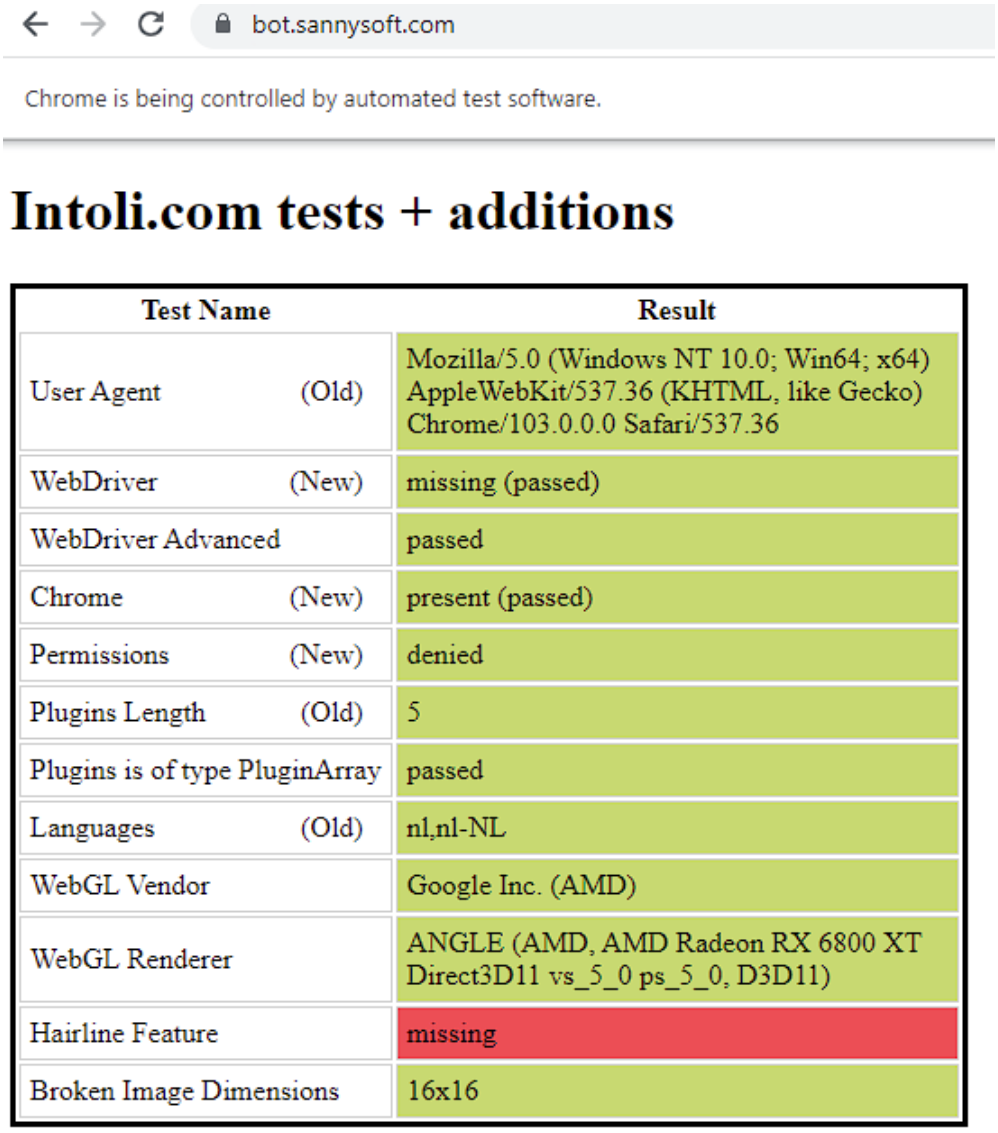


Figure 53: `https://bot.sannysoft.com/` with masking

Reducing bandwith Bandwith was possibly a limiting factor in the size of the list of websites to be visited. Several different ways were used to stop downloading images and other bandwidth hungry files while loading a webpage. Selenium-wire[52], another Selenium alternative, while also helpful in documenting requests, was very successful in blocking specific requests from the browser. Settings in Selenium were also possible to block the loading of images. Although both were successful, the reduction in bandwith usage resulted in a reduction in cookies being set and in the end no files were blocked from loading in the final crawl.

Browser side scripts Browser side scripts can be executed by Selenium making it possible to access more than with Selenium by default. There are two important situations where browser scripting was necessary. The first one for getting all cookies in the session. Using the webdriver method `driver.get_cookies()` only first party cookies can be found, but running a script in the browser all cookies can be found. The code below returns all

[52] `https://pypi.org/project/selenium-wire/`

cookies currently present in the browser session.

```
driver.execute_cdp_cmd('Network.getAllCookies', {})['cookies']
```

The information about z-indexes can also only be found running a script in the browser. The code below returns the top 30 z-index elements ranked from high to low.

```
driver.execute_script("""return \
          Array.from(document.querySelectorAll('*'))\
            .map((el) => ({zIndex: Number(getComputedStyle(el).zIndex),\
            element: el }))\
            .filter(({ zIndex }) => !isNaN(zIndex))\
            .filter(({ zIndex }) => (zIndex > 0))\
            .sort((r1, r2) => r2.zIndex - r1.zIndex)\
            .slice(0, 30);\
          console.table(data);""")
```