Automating Cookie Consent and GDPR Violation Detection

Dino Bollinger
ETH Zurich

Karel Kubicek
ETH Zurich

Carlos Cotrini
ETH Zurich

David Basin
ETH Zurich

Abstract

The European Union's General Data Protection Regulation (*GDPR*) requires websites to inform users about personal data collection and request consent for cookies. Yet the majority of websites do not give users any choices, and others attempt to deceive them into accepting all cookies. We document the severity of this situation through an analysis of potential GDPR violations in cookie banners in almost 30k websites. We identify six novel violation types, such as incorrect category assignments and misleading expiration times, and we find at least one potential violation in a surprising 94.7% of the analyzed websites.

We address this issue by giving users the power to protect their privacy. We develop a browser extension, called CookieBlock, that uses machine learning to enforce GDPR cookie consent at the client. It automatically categorizes cookies by usage purpose using only the information provided in the cookie itself. At a mean validation accuracy of 84.4%, our model attains a prediction quality competitive with expert knowledge in the field. Additionally, our approach differs from prior work by not relying on the cooperation of websites themselves. We empirically evaluate CookieBlock on a set of 100 randomly sampled websites, on which it filters roughly 90% of the privacy-invasive cookies without significantly impairing website functionality.

1 Introduction

Browser cookies are the most common method for tracking the session state of websites. While some cookies are necessary for a website to operate, such as authentication cookies that keep users logged in, the majority of cookies are used for user tracking and advertising (as we show later in Fig. 2). Despite the existence of stateless tracking techniques such as browser fingerprinting [1], stateful tracking using cookies remains the primary tracking method. In 2019, Solomos et al. report that almost 90% of all websites use tracking cookies [48], an increase from 80% observed in a study by Roesner et al. from 2012 [42].

Governments have attempted to address user tracking through regulations. In the European Union, the General Data Protection Regulation (*GDPR*) [19] and the ePrivacy Directive [18] place restrictions on personal data collection and tracking. Article 6 of the GDPR specifies that a legal basis is required for a website to collect user data, the most common basis being *consent*. Article 7 and Recital 32 specify that consent must be freely-given, unambiguous, specific, and informed. The ePrivacy Directive and Recital 30 of the GDPR specify that the consent requirements also apply to the use of cookies. Websites must thereby inform users about the purposes cookies are used for, and they must provide users with the option to deny consent for specific purposes.

The GDPR has created a demand for prepared consent solutions, from which a new "consent as a service" industry has emerged [53]. The companies offering these services, called consent management platforms (*CMPs*), provide websites with cookie banner implementations that handle the collection of consent from users [24], and offer detailed descriptions of all the purposes that cookies are used for. Unlike the simpler cookie notices, which only inform users about the mere use of cookies, CMPs promise to provide users with more control over their personal data, fulfilling the GDPR's requirements in this area. However, Kampanos et al. [30] find that in a sample of approximately 14k websites from the UK and 3k from Greece, only 44% and 48%, respectively, show a cookie banner to the user. With 90% of all websites using tracking cookies, this means that many neglect to comply with the GDPR.

Websites that use CMPs also often do not live up to their promises, with many violating even basic rules. Nouwens et al. [39] show that 88.2% out of 680 examined websites that use a CMP fail in at least one of three simple requirements, including the requirement of opt-in choices and explicit consent. Matte et al. [34] found that in a sample of 1426 selected websites, 9.89% register affirmative consent before the user makes a choice, 2.66% do not allow any cookies to be rejected, and 1.89% register positive consent even when rejected by the user. Moreover, prior work has also shown that many CMPs attempt to influence visitors into accepting

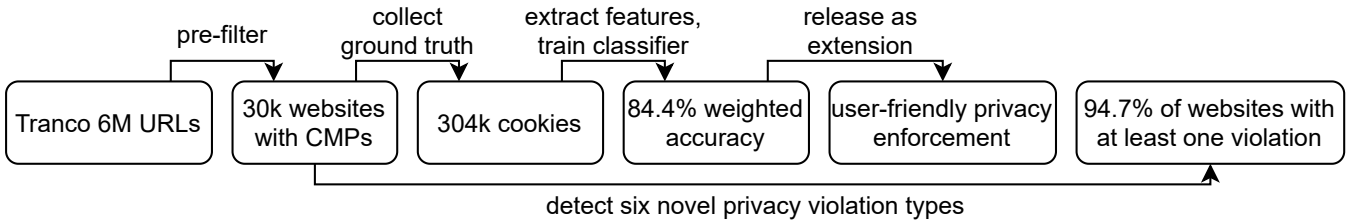

Figure 1: Overview of the process involved in our study and the (intermediate) results.

all cookies. For example, a study by Utz et al. [52] shows that 57.4% of 1000 examined websites use nudging, which involves highlighting the "Accept All" button, or hiding the option to reject consent. This trend does not appear to be changing [30], and while high-profile violations are penalized [29, 38], GDPR enforcement regarding cookies is lagging behind, as the aforementioned studies show.

Our analysis. We confirm the lack of GDPR compliance by extending and improving upon past research. We analyze the accuracy of the information displayed on cookie banners, using a dataset collected from almost 30k websites. Specifically, we identify incorrect category assignments, misleading cookie expiration times, and assess the overall completeness of the consent mechanism. We define six novel methods to detect potential GDPR violations and extend two methods used in prior works. For the selected domains, we find that 94.7% contained at least one potential violation. In 36.4%, we found at least one cookie with an incorrectly assigned purpose, and in 85.8%, there was at least one cookie with a missing declaration or missing purpose. 69.7% of the sites assumed positive consent before it was given, and 21.3% created cookies despite negative consent. Our results indicate that this problem is more severe than previously indicated.

Note that we refer to the violations found as *potential* because only a judicial ruling can provide the legal certainty as to whether they are actual legal violations. In Section 6 we argue why they should be considered violations by referring to relevant regulations and legal precedents.

Browser extension. Based on evidence from prior works and our own measurements, cookie consent practices violate the GDPR so frequently that regulatory authorities cannot hope to keep up. We therefore provide users with a tool to enforce cookie consent on their web clients, without regulatory intervention. We develop a browser extension, CookieBlock, that classifies cookies by purpose, removing those that the user rejects. In this way, users can remove over 90% of all privacy-invasive cookies, without having to trust cookie banners or CMPs. Previous attempts to provide users such control, like the P3P standard [10], failed due to a lack of willingness of website administrators to implement the functionality required. We sidestep this problem by not relying on websites' cooperation at all.

We evaluate CookieBlock on a set of 100 websites to quantify the extension's impact on users' browsing experience. CookieBlock causes no issues on 85% of the sites, minor problems involving non-essential website functions on 8%, and more substantial issues on 7%. The more substantial problems involved the user's login status being lost due to the removal of essential cookies. To resolve these problems, the user can selectively define website exemptions, and change the classification of cookies through CookieBlock's interface.

To classify cookies, CookieBlock uses an ensemble of decision trees model, trained using the XGBoost [8] library. We gathered a training dataset of cookies from 29 398 websites that display cookie banners from a specific set of CMPs. Each CMP maintains its own cookie-to-purpose mapping, which we use to define the ground truth class-labels for the cookies in our dataset.

We evaluate the model by comparing its performance with the "Cookiepedia" repository [40]. Cookiepedia assigns purposes to cookies based on their name, and was constructed manually over a span of 10 years by experts in the domain of browser cookies. We query this repository for purpose predictions and compare the results with our selected ground truth. In summary, we find that Cookiepedia achieves a balanced accuracy of 84.7%, while our XGBoost-trained model achieves 84.4%. As such, our model is competitive with the performance achieved by human experts, showing that it is possible to automatically classify cookies by purpose using only the information available in the cookies themselves.

Contributions. First, we identify inaccurate information in cookie banners, and apply this to a sample of approximately 30k websites, finding potential GDPR violations for 94.7% of them. Second, we present a machine-learning classifier that infers purposes from cookies, reaching a performance that is comparable to that of human experts. Third, we develop a browser extension that automatically removes cookies according to users' preferences, which, unlike comparable approaches, is applicable to any cookie and does not require websites to cooperate. Finally, we release our tools for web administrators, allowing them to verify and improve the cookie consent compliance of their websites.[1]

[1] An extended version of the paper, the datasets, and tools can be found at https://karelkubicek.github.io/post/cookieblock.

Organization. The remainder of the paper is structured as follows. Section 2 describes our approach to collecting cookie information from CMPs, including the purposes used for training and the data used for the website analysis. Sections 3 and 4 describe the features we extract from cookies and our classifier model. Section 5 describes and evaluates the browser extension. Section 6 presents our website analysis and demonstrates our approaches to detecting potential GDPR violations. Section 7 discusses the scope of our work and its limitations. Section 8 describes related work, and Section 9 draws conclusions.

2 Dataset collection

In this section we describe how we collected our dataset of cookies annotated with the ground-truth class-labels. This dataset is then used for both the classifier in Section 4 and the GDPR compliance analysis presented in Section 6.

We collect cookie purposes from consent management platforms (*CMPs*). In contrast to Cookiepedia, these purposes are chosen by the website administrators who control which cookies are created in the users' browsers [9,49]. As such, we collect the ground truth from parties that have full knowledge about the purposes of cookies, rather than a third-party who may not know the full context. This also allows us to assign categories to cookies that are rare and may be unknown to Cookiepedia. In Section 4.1, we show that more than 20% of the collected cookies could not be identified by Cookiepedia.

Our first step is to select CMPs that list cookies with their purposes (Section 2.1). Then, from a set of six million domains, we detect the presence of the selected CMPs (Section 2.2). For each website where a CMP is used, a web crawler gathers both the cookies declared by the CMP and the cookies that are created in the browser when interacting with the website (Section 2.3). Finally, we combine the declarations with the cookies, and obtain the training data for use with our classifier (Section 2.4).

2.1 Suitable CMPs and cookie categories

There are a plethora of CMPs, each offering its own website plugin [24]. These plugins range from simple notifications to elaborate cookie banners that allow users to choose from dozens of possible category options [31]. The purpose assignments we intend to collect can only be retrieved from a small subset of all CMPs. In this section, we describe the criteria we used to select them.

Our first criterion is that the CMP must publicly and reliably list purposes for each cookie on every website where the plugin is correctly implemented. This is essential for collecting the purpose labels that we take as the ground truth. On certain websites, CMPs may offer category choices, but they do not display which cookies belong to which category. Our second criterion is that, when this mapping exists, it must be accessible in a way that can be automatically processed, ideally hosted remotely on a server by the CMP itself. Some websites list the cookie-to-purpose mapping in their privacy policy. This is generally not useful as the HTML structure of such policies varies greatly between sites, and thus would require a specialized data extraction for each case.

Table 1: Listing of CMPs and their market share in the top 1M websites as reported by BuiltWith [6]. In the third and fourth columns, we evaluate the CMPs with respect to two criteria for collecting purpose labels.

CMP	Market share	Remote?	Labels?
Osano	2.25%	✓	✗
Cookie Notice	1.29%	✗	✗
OneTrust	1.17%	✓	✓
OptAnon	1.08%	✓	✓
Cookie Law Info	0.95%	✗	✗
Cookiebot	0.77%	✓	✓
Quantcast CMP	0.68%	✓	✗
UK Cookie Consent	0.33%	✗	✗
TrustArc	0.26%	✓	✗
WP GDPR Compl.	0.20%	✗	✗
Moove GDPR Compl.	0.18%	✗	✗
tarteaucitron.js	0.16%	✗	✗
Usercentrics	0.16%	✓	✗
CookiePro	0.15%	✓	✓
Borlabs Cookie	0.12%	✗	✓
EU Cookie Law	0.12%	✗	✓
PrimeBox CookieBar	0.09%	✗	✗
Cookie Script	0.07%	✓	✓
Cookie Information	0.06%	✓	✓
Termly	0.05%	✓	✓
Cookie Info Script	0.05%	✓	✗
Easy GDPR	0.04%	✓	✗

In Table 1 we list the CMPs with the highest market-share worldwide, as reported by the technology trend database BuiltWith [6]. For each entry, we list how suitable they are for data collection, based on our criteria. We selected the CMPs OneTrust, OptAnon, Cookiebot, CookiePro, and Termly, here displayed in boldface, which we will use for all subsequent steps of data extraction and analysis.

2.1.1 Cookie purpose categories

No law defines which set of cookie purposes the CMPs must declare. Only cookies that are strictly necessary for website operation are recognized, which as per Article 5(3) of the ePrivacy Directive do not require consent from users, and may therefore be set before interaction with the cookie banner.

Given that the categories are not regulated, this selection varies across CMPs. For instance, the Transparency and Consent Framework 2.0 (*TCF*), an industry standard defined by

Table 2: Keywords used to map purposes in CMPs to the selected categories, with the percentage of declarations matched. By * we group multiple suffixes of similar words. The "Other" category contains the cookie declarations that did not match a category, including non-English category names.

Category	Fraction	Keywords
Necessary	13.2%	essential, mandatory, necessary, required
Functional	8.7%	function*, preference, secure, security, video
Analytics	11.4%	anonym*, analytic*, measurement, performance, research, statistic*,
Advertising	60.9%	ad, advertis*, ads, ad selection, fingerprint*, geolocation, market*, personalis*, personal info, sale of data, target*, track*
Unclassified	3.9%	uncategorize*, unclassified, unknown
Other	1.9%	-

IABEurope, proposes a set of 12 purposes for cookies [17]. Others, like OneTrust, even support the definition of custom categories by the website administrator [9]. In this work, we restrict ourselves to the following four categories, as originally defined by the UK's International Chamber of Commerce [26]:

1. (Strictly) Necessary cookies, which cannot be omitted without breaking the website's main functionality, such as authentication cookies.
2. Functional cookies, which allow for website customization without collecting user data, and are not required for essential services. Examples include user-specific localization and layout customization.
3. Analytics cookies, which serve to track and analyze users' behaviors on a single domain, and are used for aggregated data collection. Google Analytics cookies are common examples from this category.
4. Advertising cookies, which serve to deliver targeted advertisements by tracking users across multiple different domains. DoubleClick or social media websites are common origins for tracking cookies.

In addition to these categories, we also identify unclassified cookies, which will be used for the analysis in Section 6. The advantages of the above four categories are that they represent an ordering from the least to most privacy-invasive types of cookies, and that they represent clearly distinct functions. This makes it easier for users to select and distinguish them.

To map the purposes listed in cookie banners to the categories we use internally, we use the keyword mapping shown in Table 2. Purposes that do not contain any of the keywords are recorded as 'Other', and are neither used for training the classifier nor for our analysis.

2.2 CMP presence crawler

After selecting which CMPs to target, we need to find domains that use these CMPs to show cookie banners. To do so, we implemented a fast website scanning procedure using the Python `requests` library to concurrently fetch the index page of multiple target websites and scan them for the presence of the desired CMP. If the CMP is used, the website is recorded as being a potential candidate for retrieving cookie labels, and otherwise, the site is filtered out.

Because of the relatively low percentage of websites that use the selected CMPs, and to maximize the amount of collected data, we initialize the presence crawl using a set of nearly six million distinct domains. Our primary source is the Tranco ranking [32] of May 5th, 2021,[2] which lists domains ranked by their estimated worldwide popularity.

Our scan was performed on an AWS EC2 server instance located in Germany, with 32 vCPUs, 64 GB of RAM, and a 10 Gigabit connection. Special care was taken to perform the scan from within an EU country, as previous works have shown that there is significant geographic discrimination with regards to GDPR enforcement. Cookie banners are generally less likely to be shown to non-EU visitors [11, 13].

In total, we find 37 587 ($\sim 0.63\%$ of 5.94M) candidate domains for the next step of our data collection process.

2.3 Scraping cookie consent information

The second stage of the data collection process is to extract the cookies and their corresponding purposes from the candidate domains. To do so, we utilize the OpenWPM framework, version 0.12.0 [16, 35], which runs multiple concurrent Firefox browser instances via Selenium. OpenWPM instruments the browser such that all cookie creations and updates are recorded. We call these cookies the *observed cookies*.

We extend OpenWPM to handle data extraction from the CMPs. The gathered information includes at least the declared name, domain, expiration time, and purpose description, as well as the purpose category of the cookie. We will refer to this data as the *declared cookies*. The exact method for retrieving the declared cookies is specific to the CMP implementation. Common to all approaches is that we retrieve the information directly from the JavaScript files that define the consent mechanism. As such, the gathered information should directly relate to which cookies are accepted or rejected depending on the users' choices in the cookie banner.

Our crawl then proceeds as follows: For each domain, after arriving on the landing page, the crawler detects which CMP is actively present on the site. Then the set of declared cookies are extracted. If this proceeds without error, the subsequent steps are intended to trigger the creation of cookies in the browser. First, the crawler consents to all cookie purposes in the cookie banner using the Consent-O-Matic

[2] Available at: https://tranco-list.eu/list/P63J/full

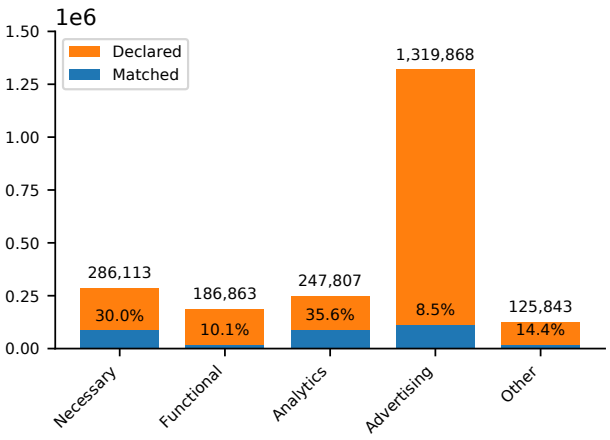

Figure 2: The total number of cookie declarations with the ratio of observed cookies that match, separated by category.

extension [27, 39]. This is required, as otherwise, the lack of consent would prevent cookies from being created. Afterwards, the browser visits random links leading to subpages of the domain, scrolling down to the bottom of each page and performing random cursor movements for each subpage. Urban et al. [51] reported that browsing subpages increases the number of observed cookies up to 36%. As a trade-off between crawling speed and the amount of collected data, we visit ten randomly selected subpages for each site.

The consent crawl was performed on the same AWS EC2 instance described in Section 2.2, and took approximately 36 hours for the $\sim$ 37.5k candidate domains. In total, we successfully extracted $\sim$ 2.2 million declared cookies from the cookie banners of 29 398 websites ($\sim$ 72 cookies per site). In addition, we extracted 602k observed cookies from those same websites ($\sim$ 22 cookies per site). We find that 81.2% of the declared cookies are third-party entries, while only 46.3% of the observed cookies stem from third-parties.

There exists a discrepancy between the number of declared and observed cookies, which we explain as follows:

Limited automated interaction with the website. Our crawler does not register an account, login or modify the website settings, which can lead to fewer necessary and functional cookies being observed.

Overabundance of declarations. CMPs may list significantly more cookies in their cookie banners than there are actual cookies to be found on the website. Papadopoulos et al. [41] find that users will encounter approximately $\sim$ 12 cookies per site. We observe a mean of $\sim$ 22 cookies, indicating that we do not observe significantly fewer cookies than the related work in the area.

2.4 Obtaining the training dataset

Our training dataset consists of the observed cookies, with purposes derived from matching cookie declarations. Each cookie is uniquely identified by its name, host, and the target domain of the crawl, and these values are used as the key to join observed and declared cookies. This produces a total of 304k cookie samples for training, of which 28.2% are necessary, 6.2% are functional, 29.0% are analytics, and 36.7% are advertising. An additional 18k cookies are unclassified, or declared a purpose that could not be assigned to any of our categories.

Fig. 2 shows the total number of declarations per category, together with the ratio of observed cookies. It is important to note that the category of functional cookies is underrepresented, which we compensate for by weighting the samples when training the classifier. Moreover, despite the overabundance of declarations, out of 602k observed cookies, only 53.6% could be matched with a declaration. This implies that there may be many cookies present on websites that are unknown to the cookie banner. We will discuss this topic in more detail in Section 6.

3 Feature extraction

Cookies have multiple attributes, including a name, domain, path, value, expiration timestamp, as well as flags such as the "HttpOnly," "Secure," "SameSite," and "HostOnly" properties. There is no straightforward relationship between these attributes and the cookies' purpose. Therefore, we extract statistically-rich, domain-specific features so that a machine-learning model can extract a potentially complex, meaningful relation from the data.

We define more than 50 feature-extraction steps that represent a cookie as a real-valued sparse vector. We provide a high-level account of these steps below. More details are provided in Appendix B and the full description is given in the extended report [3] and documentation.[3]

Top-500 most common names and domains. A very effective method for identifying a cookie's purpose is detecting whether the cookie name or its origin domain are among the most common identifiers found online. Using a representative random sample of websites from our Tranco list, we collect a ranking of the 500 most common cookie names and domains. The intuition is that web modules use first-party cookies with predefined names and purposes, such as `PHPSESSID` in the case of PHP, and that cookies originating from the same domain usually have a common purpose.

Value type, encoding, and length. Several of our features indicate the presence of specific data types in the cookie content. This ranges from scalar types such as Booleans or integers to composite types such as CSV or JSON. We also

[3]The feature documentation and classifier are available at: https://github.com/dibollinger/CookieBlock-Consent-Classifier.

record the number of entries for composite types, as well as the length of the content in bytes as ordinal features. We furthermore distinguish between decimal and hexadecimal integers, as well as base64 and URL encoded strings. The intuition is that by identifying the types of data stored in a cookie, the classifier can better distinguish which cookies are used for tracking. For example, long hexadecimal strings are more likely to be used for uniquely identifying a user than short decimals.

Dates, timestamps, UUIDs, URLs, or locale strings. These values may provide hints about the purpose of a cookie. Intuitively, dates, UUIDs, and timestamps may be used as unique identifiers for tracking, while locales and URLs are more commonly used with functional cookies, for example to alter the display language or input method.

Update Features. Cookies are dynamic, and can be frequently updated by HTTP requests or through events in JavaScript code. As such, we not only consider features for a single state of the cookie, but also for changes that occur over time. Examples are the total number of times a cookie is updated over a fixed time interval, or the edit distance between cookie updates.

Cookie entropy. The entropy of the cookie's content, for example computed using Shannon's method, can provide information about its randomness. The intuition is that tracking identifiers often include a randomly generated component and hence have high entropy, thus potentially allowing the classifier to detect tracking cookies.

Note that not all cookie features can be used in all settings. For instance, in our dataset, advertising cookies are updated more rarely than other types of cookies. While this property could be used as a feature for training, it is highly dependent on the user's browsing pattern. Any features that are based on such patterns are unreliable in the setting of a browser extension, and may cause false predictions that cannot be observed during the model validation. For CookieBlock, we therefore only use those features that are agnostic to browsing patterns. Nevertheless, such properties may still be used for offline settings with a fixed browsing behavior, such as studies involving automated web-crawlers.

4 Classification

In this section, we present the design and evaluation of our cookie purpose classifier. We first describe the baseline, which is the manually constructed repository Cookiepedia (Section 4.1). Next, we explain our choice of model (Section 4.2) and the selected hyperparameters (Section 4.3). We explain the impact of different types of misclassifications (Section 4.4), and present our model's performance, comparing it with the selected baseline (Section 4.5). Finally, by estimating the degree of noise in the data, we estimate the best possible classifier performance for this dataset (Section 4.6).

4.1 Baseline

We compare our model's performance to that of a manual classification by experts in the field. Namely, we query cookie purposes from the public cookie repository Cookiepedia [40]. Cookiepedia reportedly stores data for over 30M cookies, of which a large portion has been labelled with purpose categories. These categories match the ones we have chosen in Section 2.1.1. For our dataset, Cookiepedia provides purposes for 79.2% of the cookies.

To use Cookiepedia as a classifier, we query it for each cookie name in our dataset and obtain the corresponding purposes from the repository. These purposes are then compared to the class labels we collected from the CMPs. To validate Cookiepedia as a classifier, we split the cookie dataset into 5 equally-sized chunks and compute the average accuracy, precision, and recall. In Table 3 we present the results.

Our measurements show that Cookiepedia achieves a mean balanced accuracy (i.e., macro-recall) of 83.4%. It achieves a high precision for both necessary and advertising cookies, but has particularly low precision for functional cookies. This can be explained through the class imbalance we find in the validation data. Due to the low number of samples for the functional ground truth, any error that assigns this category to other cookies will have a much greater effect on the precision of this class than it would have for the other categories.

4.2 Model selection

Our chosen model for the task of classifying cookies are ensembles of decision trees. We train them using the XGBoost library [8], which uses a sparsity-aware gradient tree boosting method developed by Chen and Guestrin. We use boosting because ensembles of decision trees can be as competitive as neural networks and have achieved top performance in several machine-learning competitions and benchmarks [20, 45, 54].

In the setting of multi-class classification, XGBoost creates a classifier model with a forest of decision trees for each purpose class. Given a sparse input vector representing a cookie, the model produces a probability for each purpose that indicates how likely the cookie belongs to it. Using a Bayesian Decision function, we transform these probabilities into a discrete prediction. For our evaluation, we apply a simple `argmax` decision, i.e., the purpose with the highest probability is chosen as the prediction.

4.3 Training parameters

The dataset we use consists of 304k labeled cookies, of which 277k are used for training. The 27k cookies we filter out are cookies created by CMPs to track users' interaction with the cookie banner. With this filtering, we aim to remove training bias as these cookies are always present on the sites we crawled, but are not common outside the chosen websites.

Table 3: Performance metrics for the Cookiepedia lookup. Evaluated using 277k cookies, as an average over 5 folds.

Cookiepedia	Necessary	Functional	Analytics	Advertising
Precision	94.5% ±0.2%	38.1% ±0.6%	84.2% ±0.2%	94.9% ±0.1%
Recall	88.5% ±0.1%	78.7% ±1.1%	93.0% ±0.1%	79.0% ±0.2%

Cookie coverage: 79.2%
Accuracy: 86.1% ± 0.1%
Macro-recall (balanced accuracy): 84.7% ± 0.3%

Table 4: Performance metrics of the XGBoost classifier in categorizing cookies, trained on 277k samples and evaluated with 5-fold cross-validation.

XGBoost	Necessary	Functional	Analytics	Advertising
Precision	87.3% ±0.2%	52.9% ±0.5%	89.8% ±0.3%	93.6% ±0.2%
Recall	81.7% ±0.5%	76.3% ±0.5%	89.7% ±0.2%	89.8% ±0.3%

Cookie coverage: 100%
Accuracy: 87.2% ± 0.23%
Macro-recall (balanced accuracy): 84.4% ± 0.27%

To find good hyperparameters, we applied a randomized grid-search with 5-fold cross-validation. The performance of each model is validated using the multi-class cross-entropy loss, as well as the balanced accuracy, due to the training dataset being imbalanced. The most impactful parameters were the learning rate and the maximum tree depth, for which we selected a rate of 0.25, and a depth of 32, respectively. Further increasing the depth leads to a decrease in the validation performance. We trained each model for a maximum of 300 boost rounds, with early stopping after 20 rounds with no increase in validation score. For the final model, there are 12 to 29 trees per forest, with the average size being 22 trees. The complete set of parameters is shown in the Appendix in Table 6.

4.4 Impact of misclassifications

As mentioned in Section 2.1.1, our selected purpose categories can be interpreted as an ordering, with necessary being the least and advertising the most privacy-invasive. Using this ordering, a misclassification of a functional cookie into the necessary category has reduced privacy impact, as the functional cookie is close in the ordering, and unlikely to be used for user tracking. A wrong assignment of an advertising cookie to necessary represents a greater privacy threat as these categories are far apart in the ordering, with tracking cookies potentially being unconditionally permitted.

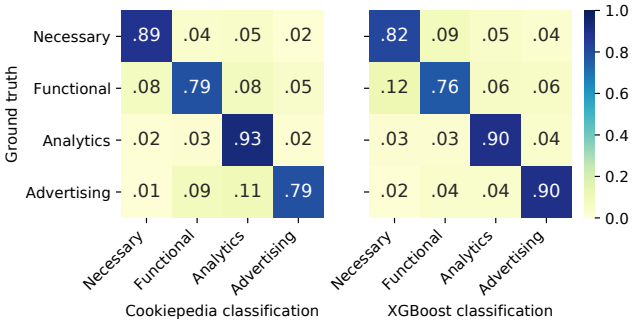

Figure 3: Confusion matrices of the Cookiepedia baseline and XGBoost. Each entry C_{ij} shows the ratio of cookies with ground truth i that were assigned purpose j.

Similarly, we also consider the potential of websites breaking due to misclassifications. When a necessary cookie is predicted as advertising, and thereby removed, it may break an essential service on the site, and drastically reduce the quality of the user experience. Assigning the class functional to a necessary cookie has a reduced impact as users are less likely to reject this purpose due to it being less privacy-invasive.

The probability with which advertising cookies will evade detection can be identified using the recall metric of the advertising class. The potential to break essential functionality on websites can be found in the recall of the necessary category. The closer either performance metric is to 1, the lower the privacy threat, respectively the less likely a website is to break.

4.5 Evaluation

Fig. 3 compares the performances of XGBoost and Cookiepedia. Table 4 presents the performance metrics for our XGBoost model. We discuss them next.

XGBoost attains higher privacy protection. In accordance with Section 4.4, we first consider the potential privacy protection through the recall of the advertising category. Here, the recall measures the fraction of advertising cookies correctly identified as advertising by our classifier. XGBoost's recall is almost 9% higher than that of Cookiepedia. In Fig. 3, we see that Cookiepedia's misclassifications in this regard occur mainly because it assigns advertising cookies to the analytics or functional class.

XGBoost preserves necessary and functional cookies. We consider the potential for websites breaking. The recall for necessary cookies for the XGBoost classifier is 81.7%, almost 7% lower than what Cookiepedia achieves. For functional cookies, we have a recall of 76.3%, roughly 2% lower than Cookiepedia. Fortunately, as we see in Fig. 3, most of the misclassifications of necessary are assigned to the functional purpose, and vice-versa. Therefore, if users accept both necessary and functional, the extension will retain approximately 91% of the necessary and 88% of the

functional cookies. We verify this empirically in Section 5.3.

XGBoost is as competitive as human experts. Our automated XGBoost model performs very similarly to the manually curated Cookiepedia in the remaining metrics. Both have a reduced precision and accuracy in functional cookies, which occurs due to the class imbalance. Additionally, both achieve a high recall for the analytics class, with XGBoost achieving an improved precision by more than 5%.

To summarize, Cookiepedia achieves a balanced accuracy of 84.7% on our dataset when queried for each cookie name. Our automated, XGBoost-trained classifier achieves a balanced accuracy of 84.4%, thus attaining a performance that is comparable to the performance achieved by human experts. While Cookiepedia is more accurate in the necessary category, XGBoost performs better with advertising cookies. Our deficit in necessary cookies can be counterbalanced by using an alternative Bayesian cost function, which penalizes misclassifications of necessary cookies more strongly than others. We can also provide users of CookieBlock with ways to correct the classification, which we describe in Section 5.

Finally, the number of cookies that Cookiepedia can classify is limited. For our dataset, Cookiepedia is able to provide a category for 79.2% of the cookies, while our classifier can predict a class for every cookie.

4.6 Performance upper bound

In this section, we try to estimate the theoretically best classifier performance on our dataset. The cookie labels we collected are noisy, as different websites can use the same third-party cookie, but they do not necessarily agree on its purpose. This means that it is impossible to achieve 100% accuracy on this dataset, as some cookies will be indistinguishable despite differing purposes. To estimate the percentage of cookies in the dataset for which this is the case, we collect the majority class for each third-party cookie name and domain, and compute the percentage of cookies with a deviating class. This gives us a lower bound of 7.2% of labels that are noise among the third-party cookies.

If we assume that the noise of the first- and third-party cookies is similar, we can conclude that we have an upper bound of roughly 92-93% in overall accuracy. With an overall average accuracy of 87.2%, we argue that our classifier is close to the best possible performance on this dataset.

5 Browser extension

In this section, we describe the design and implementation of CookieBlock.[4] It is an extension for Firefox and Chromium-based browsers that automatically classifies cookies into purpose categories, and allows users to deny consent for selected purposes. By using the classifier described in Section 4, we provide users with a tool to enforce the GDPR and protect their own privacy when handling cookies.

We first discuss the goals and features of CookieBlock (Section 5.1). Then we present its design and implementation (Section 5.2). We conclude the section with an empirical evaluation on a set of 100 websites that estimates how CookieBlock affects users' browsing experience (Section 5.3).

5.1 Goals and Features

The objective of CookieBlock is to give users control over their privacy, a practice that is neglected by the majority of websites. Table 1 indicates that out of the top 1M websites, only an accumulated total of 3.5% use CMPs providing cookie consent choices, and many of those that do deceive users either by dark patterns, as shown by Nouwens et al. [39], or by providing wrong information, as we show in Section 6. Hence CookieBlock provides users with a means to control their cookie consent on any website they visit, without the risk of being deceived. CookieBlock offers the following features:

- *User-defined cookie policy.* CookieBlock's central feature is that users specify which of the four categories in Section 2.1.1 they give or deny consent to. All cookies belonging to a purpose for which consent was denied are then removed from the browser's storage.
- *Domain exceptions.* For domains that the users trust, they can define an exception. The extension will not remove any cookies originating from exempted domains, regardless of their purpose.
- *Custom cookie categories.* Users can define their own cookie categories, which can be used to correct individual mistakes made by the classifier.

Note that while CookieBlock imitates the behavior of a CMP, it is not intended to interact with or remove the cookie banners shown on websites. This function is already fulfilled by existing browser extensions, such as Consent-O-Matic [27], which can be used in conjunction with CookieBlock. CookieBlock also does not act as a replacement for the cookie banner in the legal sense, and its use is not a justification for websites to skip the gathering of user consent.

5.2 Design and implementation

CookieBlock is built using the WebExtensions API, and supports Firefox as well as Chromium-based browsers. An overview of its design is given in Fig. 4.

5.2.1 Background process

On initialization, CookieBlock begins listening for cookie events. When a cookie is created or updated, the cookie's current state is appended to a local cookie history (1), and the full list of previous updates for that cookie is retrieved (2). This

[4] Available at https://github.com/dibollinger/CookieBlock.

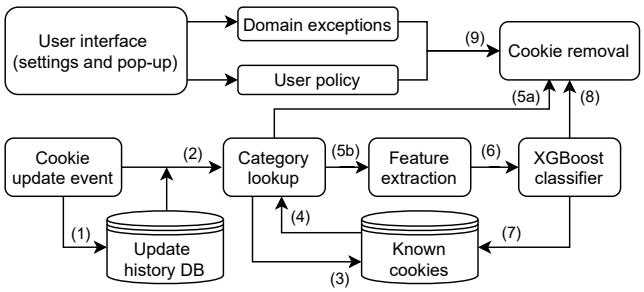

Figure 4: Outline of CookieBlock's design.

history allows CookieBlock to track the evolution of a cookie over time, a property which is used in the feature extraction.

Afterwards, CookieBlock checks its storage to determine whether the cookie has been encountered recently or whether it has been assigned a pre-defined category (3). In this case, it retrieves the existing purpose label (4), and skip directly to the policy enforcement step (5a). If the cookie does not have an existing label stored, then we proceed to the feature extraction (5b). This transforms the cookie object into a sparse vector representation (6). It then runs the precomputed XGBoost model on this vector input, which predicts a purpose label for the cookie. The predicted label is then cached in the extension storage for a short duration (7). Finally, the predicted label is passed on to the policy enforcement procedure (8), which decides whether to keep or remove the cookie.

To decide whether to keep or remove a cookie, CookieBlock takes into account the user's cookie policy and domain exceptions (9). If the origin domain of the cookie matches a domain in the set of domain exceptions, the policy enforcement will always retain the cookie.

5.2.2 User interface

The user interface is structured into four distinct components:

First-time setup. The first-time setup page of the extension allows the user to define a user-policy, and requests consent to the collection of a local cookie history. This is the minimal setup required to initialize the extension.

Settings page. The settings page allows users to change their consent preferences at any time and add individual website exceptions.

Toolbar popup. The toolbar popup offers a quick method to pause the cookie removal and to add an exception for the domain in the address bar.

Cookie configuration. The cookie configuration page allows users to define custom categories for previously encountered cookies and to correct misclassifications.

For both the settings page and the first-time setup, CookieBlock allows the user to consent to the functional, analytics, and advertising purposes. The necessary category cannot be rejected as doing so would break websites.

We designed the interface to be simple to use and unobtrusive. Unlike cookie banners found on different websites, CookieBlock requires only a single setup, after which the users' cookie preferences will be enforced on all websites. This prevents the issue where privacy is neglected due to user fatigue or annoyance from cookie banners [2, 28].

5.2.3 Cookie update history

As described previously, CookieBlock collects a cookie update history. This allows it to track how cookies change over time, enabling predictions based on these differences. It also allows CookieBlock to remember past purpose assignments by recognizing which cookies have been encountered before.

Since this cookie history may contain potentially sensitive user information, including information about the browsing history and authentication tokens, the history is kept local to the browser extension at all times. In addition, CookieBlock asks the user to opt-in to the collection of this history at setup time. If rejected, CookieBlock can still classify cookies, but it will not be able to remember previous labels or extract features from past updates, which may reduce its accuracy.

5.2.4 Cached purposes

CookieBlock caches labels for a short period after a prediction is made. This minimizes browser slowdown in case a website continuously regenerates cookies after they have been removed. After the grace period expires, the cookie will be reclassified using newly collected cookie updates.

5.3 Empirical evaluation

As noted in Section 4.5, our classifier has a recall of 81.7% on necessary cookies, meaning that potentially every fifth cookie required for the operation of website could be misclassified. Since CookieBlock uses the computed model as a predictor, many necessary cookies may inadvertently be removed, causing websites to malfunction. However, due to the noise in the dataset, it is unclear how severe this issue is in practice.

To quantify the impact CookieBlock has on the browsing experience, we manually visit and examine a sample of 100 websites for possible malfunctions. We acknowledge that this evaluation is limited in that it does not constitute a full usability study. However, because the extension acts as a background process, it should ideally require very little interaction with the user. We therefore focus on evaluating whether a website breaks due to misclassification, which is the critical aspect of usability in this case.

We randomly sample websites from the Tranco list from Section 2.2 using an exponential distribution. This allows us to examine both popular as well as niche websites. Furthermore, this website selection is not restricted to those that use specific CMPs.

We use a clean installation of CookieBlock, configured to allow necessary and functional cookies, which is the recommended setup. For each website, we attempt to make use of its primary services as best as possible, recording any defects we encounter in the process. We also attempt to change website settings, such as the language or style, and we attempt to register an account and perform the login procedure where available. Finally, we also interact with and close cookie banners, recording whether any appear again on page reload. A reappearing cookie banner can be very annoying for the user, but it does not prevent the site's use, and therefore these are likely misclassified functional cookies. If we encounter any unexpected behavior, we determine whether this was caused by CookieBlock by disabling the cookie removal.

Our results show that out of the examined 100 websites: 85 showed no obvious malfunctions, 7 had a cookie banner that reappeared because of CookieBlock, 7 showed an authentication failure where the user was immediately logged out, and in one case, we could not change the website language. As such, the rate of serious defects is less severe than expected. Furthermore, all issues were resolved by defining an exception for the current site, or by correcting the cookie's assigned purposes in the extension interface.

We also measured the time it takes for CookieBlock to make a policy decision for a cookie. We ran CookieBlock on the Firefox browser on Linux, and it processed a total of 5561 cookies observed from real-world websites. Each decision took on average $\sim$ 20ms, with a maximum time of 4.3 seconds. This outlier was caused by asynchronous execution in the browser. The Firefox browser also reports a "low" energy impact for the extension.

6 Observed violations

Article 7 and Recital 32 of the GDPR require that consent must be freely given, specific, informed, and unambiguous; hence any cookie banner that displays misleading or false information may violate the law. In this section, we present an analysis on the data displayed by selected suitable CMPs, performed on a dataset of cookies from 29 398 websites, the collection of which we described in Section 2. For these websites, we assess the correctness of the cookie-to-category assignments shown on the cookie banner, the claimed expiration time of cookies, as well as the completeness of the cookie banner. These approaches encompass six novel analysis methods not explored in prior work.

Additionally, we extend the studies of Nouwens et al. [39] and Matte et al. [34] by making use of the cookie purposes collected from CMPs. Namely, we analyze whether websites assume implicit cookie consent or respect the users' consent choices. We accomplish this by observing which types of cookies are set in the browser.

In summary, out of 29 398 websites, 94.7% contain at least one issue, while 77.3% have at least two. A detailed breakdown of the results is given in Figs. 5 and 6. The following subsections will elaborate on the analysis in greater detail.

6.1 Incorrect cookie purposes

The CMPs we selected in Section 2.1 declare purposes for the corresponding cookies. We inspected the accuracy of these declarations using several complementary methods.

Incorrect purpose for well-known cookies. Google Analytics cookies, such as `_ga`, `_gat`, and `_gid`, occur commonly throughout the web and have a well-known purpose. There nevertheless exist numerous websites that do not declare these cookies as analytics. In the case of Google Analytics, 8.2% of the 29 398 examined websites assign an incorrect purpose to these cookies. Moreover, 2.7% of all websites declare at least one GA cookie as necessary, which the EU Court of Justice previously ruled to be a violation of the GDPR, as decided on the Planet49 case [29].

Incorrect purpose based on the majority opinion. In the collected dataset, we observe that for identical third-party cookie identifiers, different domains may disagree on the purpose. We used this fact to estimate a performance upper bound for the classifier in Section 4.6. Here, we use it to detect outlier purpose assignments, which likely indicates an incorrect declaration. We find that 30.9% of websites contain at least one third-party cookie with a purpose that disagrees with a corresponding two-thirds majority.

This serves as a lower bound on the number of potential violations. In the event where the majority class is false, the number of potential violations would be even greater. Because this is only a lower bound, each case detected using this method requires manual analysis to determine whether it constitutes a true misclassification.

Cookies with multiple labels. An ambiguity occurs when the same website labels a cookie multiple times for different or even contradictory purposes. We observe this in 2.3% of the examined websites. This ambiguity may deceive users, as it is not well-defined whether rejecting only one of the purposes suffices to prevent the cookie's creation. In practice, we observed websites creating cookies with one purpose accepted and one rejected. Moreover, in 0.7% of the sites, the cookie is declared both as necessary and another purpose, which means that these cookie cannot be rejected at all.

6.2 Unclassified and undeclared cookies

The CMPs we target in our study offer a cookie scan service that detects cookies on a website and suggests purposes based on a database lookup. Those cookies that cannot be annotated in this fashion must have their purposes assigned manually by the site administrator [9, 49].

We find two problems with this process. First, when the web administrator neglects to assign a purpose, the cookie becomes unclassified. Second, when the CMP scan fails to

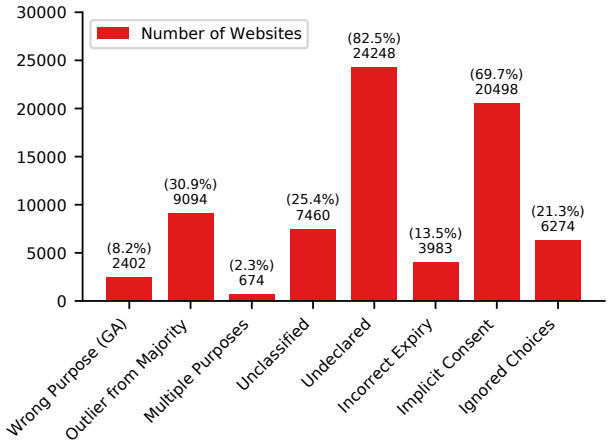

Figure 5: Number of websites that show the respective type of violation. The first six are novel and have not been explored in prior work.

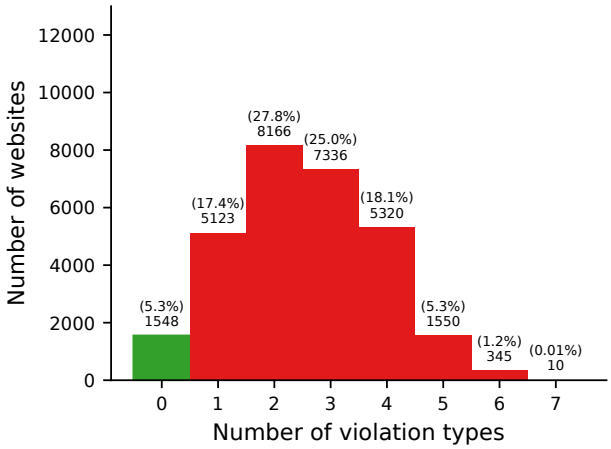

Figure 6: Histogram that shows the distribution of violation types per site. This does not include repetitions of a single type. The green bar represents the compliant websites.

detect cookies, or the cookies are added after the scan, those cookies are undeclared and are missing from the cookie banner. The website's visitor can reject neither the undeclared nor unclassified categories, which means that the consent is both uninformed and not freely given.

Unclassified cookies. We find unclassified cookies in 25.4% of the examined websites. These websites contain on average 11 unclassified cookies. Surprisingly, we find 45 websites that contained more than 200 unclassified cookies.

Undeclared cookies. We detect undeclared cookies by identifying which observed cookies do not have a matching declaration. When matching on name and domain, we find undeclared cookies in a staggering 82.5% of the examined websites. Of the 496k cookies, 40.2% were undeclared. Similarly to unclassified cookies, we find 71 websites with more than 100 undeclared cookies.

6.3 Incorrect expiration time

Article 13(2)(a) of the GDPR requires websites to declare the expiration time of personal information. The EU Court of Justice in the Planet49 case decision [29] clarifies that this also applies to cookies. We therefore compare the true expiration time of the observed cookies with that of the corresponding declaration. If the true expiration time is 50% longer than the declaration states, with a minimal difference of one day as threshold, then we consider it a potential violation. Additionally, we also identify all persistent cookies that are declared as session cookies, and vice-versa. In total, 9.1% of all sites show at least one expiration time discrepancy, 3.8% declare a persistent cookie as a session cookie, and 3.1% declare a session cookie as persistent.

6.4 Extension of previous approaches

The following two approaches extend methods defined in the works of Matte et al. [34] and Nouwens et al. [39]:

Cookies set prior to user's consent. Article 5(3) of the ePrivacy directive states that only necessary type cookies may be created prior to the user's interaction with the CMP. By crawling the website without interacting with the cookie banner, we inspect if websites set any cookies with a purpose that is not declared as necessary. We find that 69.7% of the examined websites set such cookies, and hence use implied consent. This aligns with the results by Nouwens et al. [39], who found that 67.6% of 680 sites used implicit consent. In contrast, Matte et al. [34] only found implicit consent on 9.9% of 1426 analyzed websites.

Cookies set despite negative consent. Using the Consent-O-Matic browser extension [27], we reject all purposes other than necessary. We then verify that the recorded consent status of the CMP is indeed negative, and identify which of these websites still set non-necessary cookies. We do this only for Cookiebot, as for this CMP we can verify whether the cookie banner was interacted with. For the 9446 Cookiebot domains, 66.4% set at least one cookie with a rejected purpose. This corresponds to 21.3% of the 29 398 websites we examined. However, we expect that other CMPs behave similarly, and that the total ratio is higher. For comparison, Matte et al. [34] found that 5.3% of 508 analyzed websites store user's positive consent to categories that the user rejected.

6.5 Summary

Fig. 5 summarizes the number of potential violations for each of the types we described above. In Fig. 6, we present how many different violation types are present on websites in our

dataset. The histogram shows that the median number of violations is 2 and the average is 2.5.

The first six bars in Fig. 5 represent analysis methods that, to the best of our knowledge, have not been explored in prior works. The latter two extend analyses previously performed by Nouwens [39] and Matte [34], who examined these issues by analyzing the consent string registered by CMPs. Our approach is more fine-grained and direct, as we directly detect the cookies created in the user's browser, based on the purposes declared in the cookie banner. Our sample size of websites is also much larger than in both their works.

For the case of unclassified and undeclared cookies, we believe that the issues usually stem from neglect rather than malice. The cause is likely the lack of enforcement and web administrators who are not sufficiently familiar with the legal requirements. This can be addressed with the methods described in this paper. Regulatory authorities can improve enforcement of the GDPR by automatically determining which websites violate the law. Moreover, CookieBlock and the corresponding web crawler can help web administrators inspect the compliance of their website by detecting undeclared cookies, and predicting purposes for currently unclassified cookies.

7 Limitations

Given the involved complexity of collecting the training dataset and the application of machine learning, we are aware of the following limitations of our approach.

The training dataset might be biased. There are several reasons why our training dataset might be biased. First, we collect cookies only from websites that use the services of a CMP and which assign purposes to individual cookies. Cookies used by such websites can differ from those that are found on generic websites. Second, our crawling underrepresented the functional cookies, which led to a decreased precision for this class. With a more advanced crawler or manual cookies collection, we might improve the classifier performance and remove potential bias. Third, the features we collect in an automated crawl can differ from the features resulting from users browsing websites. To address this, we remove features that depend on browsing patterns, such as cookie updates. However, if the websites can detect our crawler as a bot, they can serve different data to the crawler than to a real user. Lastly, the model should be kept up-to-date, otherwise the validity of the training data can become outdated. We address this by simplifying the process for collecting the training data as well as the training itself.

The cookie removal may not always protect users. CookieBlock removes cookies after their creation, rather than blocking the requests that spawn them. This may not be sufficient to prevent cookies from fulfilling their purpose. We rarely observe cookies that are created and removed by the website more quickly than the $\sim$ 20ms required to process the cookie by CookieBlock. One example is the cookie `GoogleAdServingTest`, which serves to record which advertisements have been displayed to the user. Fortunately, such cookies are rare.

This limitation exists because it is not possible to prevent cookie creation within the WebExtension API. We can only remove a cookie after it was already stored in the browser. Ideally, our work inspires web browser developers to allow extensions to prevent cookies from being set, or even add "purpose" as a new cookie parameter. This parameter would also address the limitation of machine learning imprecision, but our classifier would still be useful to bootstrap the cookie classification for web administrators.

We do not consider adversarial websites. We did not address the possibility that websites could alter the content of their cookies specifically to counteract the cookie policy enforcement by CookieBlock. For example, an adversarial website could change the cookie's name to a randomly generated value, use a proxy domain to alter the cookie's host field, or obfuscate cookie's content. Still, it is easier to use other tracking technologies that do not involve cookies, which we do not consider in this work. However, some websites, such as those that use the CookieBot CMP also declare other tracking resources like localStorage or tracking pixels. Therefore, it is possible to extend CookieBlock with a classification of these alternative tracking methods. We have not done this because these declarations are rare and would require a completely different feature-engineering and classification approach.

8 Related work

Cookie classification. In [25], Hu et al. propose a cookie purpose classifier that uses a Multinomial Naive Bayes model, which takes as input n-gram tokens extracted only from the cookie names. They train their model on 11.5k cookies with ground-truth labels taken from Cookiepedia, and state an F1-score of 94.6%. They also report a confusion matrix for one fold, which achieves an F1-score of only 86.7%.

Their work shares similarities with ours, but both works were developed simultaneously, with neither party being aware of the other. Our approach differs in two main respects. First, rather than using just the cookie name as a feature for training, we extract features from all cookie properties, including those that are observed between cookie updates. While the cookie name is simple to alter, the value and domain are restricted by the implementation requirements, e.g., a tracking cookie requires a minimum amount of entropy. This fact makes spoofing Hu et al.'s model by an adversarial web developer much easier than our model. Moreover, their model cannot distinguish cookies with the same name (e.g., user_id) but with different purposes and originating from different domains. Calzavara et al. [7, Sec. 5.2.1] showed that many cookies use naming conventions for unexpected purposes, which is not reflected by Cookiepedia's use of a single classification.

Secondly, our model is trained on ground truth collected

from CMPs, while Hu et al. use ground truth labels collected from Cookiepedia. We elaborate on the advantages of our choice in Appendix A. Their classification task is also not affected by noise, which allows for a higher theoretical performance bound. This is because Cookiepedia will always report the same category for the same cookie name. By replacing the CMP labels with Cookiepedia labels on our dataset, our model accuracy increases from 87.2±0.23% to 89.2±1.3%. We provide additional details on these results in Appendix A.2.

Calzavara et al. [7] used ML models to detect authentication cookies. They used a training sample of 2.5k cookies with 332 authentication cookies. They propose feature extraction from both the cookie name and other attributes, such as the entropy and the length of the cookie value, the expiry or whether the cookie is http-only. All their features or equivalent ones are included in our feature extraction. Their binary classification achieves an F1-score of 83% in classification tailored towards the high recall of 89%.

Website privacy enforcement tools. There exists various proposed privacy enforcement tools by academia and industry. The Platform for Privacy Preferences (*P3P*) [10] is a framework for visualizing privacy policies on the client-side and enforcement of user preferences on the server-side. Although it was proposed as a W3C standard, it was never widely adopted, and Google and Facebook even bypassed P3P [5]. Another project, now discontinued because of lack of interest by websites, is the "Do Not Track" HTTP request header [21, 46]. Unlike these attempts to protect user privacy, the success of CookieBlock does not depend on the cooperation of the visited websites.

Major browsers are addressing user tracking by various means. Firefox introduced "Enhanced Tracking Protection" with controls such as blocking social-media tracking cookies [36] and "Total Cookie Protection" for partitioning third-party cookies per origin websites [37]. The Chromium Project proposed "Privacy Sandbox" [12] that plans to deprecate third-party cookies by 2022. These projects face similar issues as our project, in that they also need to white-list necessary third-party cookies, such as those for single sign-on. Compared to these efforts by browsers, CookieBlock also allows blocking for first-party cookies, such as Google Analytics.

The browser extensions Consent-O-Matic [27] and Cliqz-Autoconsent [33] have similar goals as CookieBlock. They enforce users' cookie policies on websites by automating interaction with the CMPs. However, they are limited to websites that use supported CMPs, and they also depend on the website's honesty to follow the consent. In Section 6 we showed that dependence on the CMP's implementation still leads to multiple potential privacy violations. By being universally applicable, CookieBlock can provide stronger privacy guarantees.

Another privacy enhancing browser extension is Privacy Badger [15]. This extension logs third-party requests that perform fingerprinting or set cookies containing enough entropy to be used for tracking. When such tracking information is found in multiple websites' requests, Privacy Badger adds this third party to a blocklist. The construction of the blocklist used to happen individually in browsers, but this is prone to fingerprinting [14]. Hence Privacy Badger developers bundle the same blocklist constructed from an automated crawl to all users. CookieBlock focuses only on cookies, and it blocks them individually, compared to Privacy Badger which blocks the whole domain once it is detected to perform tracking. Unlike CookieBlock, Privacy Badger cannot prevent tracking using first-party cookies.

Studies of cookie consent compliance. Researchers are continuously scrutinizing cookie consent compliance. Kampanos et al. [30] analyzed 17k websites in the UK and Greece and found that roughly 45% have a cookie banner. They also find that most of the websites nudge users into accepting all cookies. Matte et al. [34] inspected 1426 websites that use CMPs that are part of IABEurope's Transparency and Consent Framework. They find that 10% of these websites set consent before user action, and 5% do not respect the choice to opt-out. In addition, Matte et al. develop a browser extension called "Cookie Glasses" that detects dishonest CMP implementations. Trevisan et al. [50] found that 49% of the inspected 36k websites set profiling cookies before users consent to them.

The study by Santos et al. [44] provides extensive legal background on cookie consent in EU jurisdictions. They define 17 requirements on valid cookie consent, some of which we inspected in our study.

There are several analyses of dark patterns of cookie consent notices, often supplemented with a user study. Nouwens et al. [39] found that almost 90% of an examined 680 websites using supported CMPs do not meet the GDPR requirements for valid consent. A user study by Utz et al. [52] inspected how the design of consent popups from 5k websites nudge users into uninformed consent. Since the field of the dark patterns is very active, we list further studies [4, 22, 23, 43, 47], and refer the reader to Dark Patterns workshop at ACM CHI.

9 Discussion and conclusions

Many websites do not give users a choice over which cookies are collected, despite the GDPR and ePrivacy Directive requirements. Multiple prior studies report on this, and we contribute to this analysis by showing that even from the websites providing choices, the vast majority, namely 94.7%, contain at least one potential violation. This situation cannot be resolved through new regulations alone, such as the planned ePrivacy Regulation, as it is mostly enforcement that is significantly lacking behind.

We address this situation with CookieBlock, which enforces the user's cookie policy on the client-side. It removes cookies based on purposes assigned by a classifier model that was trained using the XGBoost library, which achieves a performance close to that achieved by human experts. Unlike

previous, now deprecated, standards like P3P and "Do Not Track," CookieBlock does not depend on the cooperation of the websites. Beyond this, the extension and the violation detection methods can provide regulatory agencies with an automated procedure for violation detection and help them to enforce compliance to privacy regulations.

In an ideal world, CookieBlock would not be needed. Future privacy regulations could request the browser vendors and the World Wide Web Consortium to extend cookie headers with a "purpose" flag as a new attribute, which would allow integration of the act of providing consent to cookies into the browser, and the cookie banner could be made obsolete. If the use of said flag were required, then users could get the privacy protection they deserve by law. Our classifier would be helpful to bootstrap this change, as it could predict a purpose for any cookie that does not have one specified. This would help the web transition from the status quo to a future with transparent cookie declarations. Until major browser vendors take action, CookieBlock can help enforce users' cookie policies on any website, even for users outside the European Union.

References

[1] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. FPDetective: dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1129–1140, 2013.

[2] Alessandro Acquisti and Jens Grossklags. Privacy and rationality in individual decision making. *Security & Privacy, IEEE*, 3:26 – 33, 02 2005.

[3] Dino Bollinger. Analyzing cookies compliance with the GDPR. Master's thesis, ETH Zurich, 2021. https://doi.org/10.3929/ethz-b-000477333.

[4] C. Bösch, B. Erb, F. Kargl, Henning Kopp, and Stefan Pfattheicher. Tales from the dark side: Privacy dark strategies and privacy dark patterns. *Proceedings on Privacy Enhancing Technologies*, 2016:237 – 254, 2016.

[5] Jon Brodkin. Google tricks Internet Explorer into accepting tracking cookies, Microsoft claims, February 2012. https://bit.ly/3exMVMp; Accessed on 13.03.2021.

[6] BuiltWith.com. Privacy compliance usage distribution in the top 1 million sites, October 2020. https://web.archive.org/web/20201021075918/https://trends.builtwith.com/widgets/privacy-compliance/.

[7] Stefano Calzavara, Gabriele Tolomei, Andrea Casini, Michele Bugliesi, and Salvatore Orlando. A supervised learning approach to protect client authentication on the web. *ACM Trans. Web*, 9(3), June 2015.

[8] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[9] CookiePro. Legacy article – categorizing cookies. https://web.archive.org/web/20210208155826/https://community.cookiepro.com/s/article/UUID-6f01b88c-0440-0642-3610-819c6ca0f7c4, Jan 2021. Accessed on: 2021.02.08.

[10] Lorrie Faith Cranor. P3P: Making privacy policies more useful. *IEEE Security and Privacy*, 1(6):50–55, November 2003. https://www.w3.org/TR/P3P11/.

[11] Adrian Dabrowski, Georg Merzdovnik, Johanna Ullrich, Gerald Sendera, and Edgar Weippl. Measuring cookies and web privacy in a post-GDPR world: Methods and protocols. In *International Conference on Passive and Active Network Measurement*, pages 258–270. Springer, 03 2019.

[12] Chromium Developers. Chromium projects – the privacy sandbox, June 2021. https://www.chromium.org/Home/chromium-privacy/privacy-sandbox.

[13] R. V. Eijk, H. Asghari, Philipp Winter, and A. Narayanan. The impact of user location on cookie notices (inside and outside of the European Union). In *Workshop on Technology and Consumer Protection (ConPro'19). IEEE*, 2019.

[14] Electronic Frontier Foundation. Privacy Badger is changing to protect you better. https://www.eff.org/deeplinks/2020/10/privacy-badger-changing-protect-you-better; Accessed on 2021.09.11.

[15] Electronic Frontier Foundation. Privacy badger. https://privacybadger.org/, 2019.

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

[17] IAB Europe. IAB Europe Transparency and consent framework policies, May 2021. https://web.archive.org/web/20210520213158/https://iabeurope.eu/iab-europe-transparency-consent-framework-policies/.

[18] European Parliament, Council of the European Union. Directive 2002/58/EC of the European Parliament and

of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications), July 2002. http://data.europa.eu/eli/dir/2002/58/oj; Last accessed on: 2021.02.06.

[19] European Parliament, Council of the European Union. Regulation (EU) 2016/679 Of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), April 2016. http://data.europa.eu/eli/reg/2016/679/2016-05-04; Last accessed on: 2021.02.06.

[20] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[21] Github. "WG closed – w3c/dnt@5d85d6c". https://github.com/w3c/dnt/commit/5d85d6c3; Accessed on 2021.14.03.

[22] Paul Grassl, Hanna Schraffenberger, Frederik Zuiderveen Borgesius, and Moniek Buijzen. Dark and bright patterns in cookie consent requests. *PsyArXiv*, 2020.

[23] Philip Hausner and Michael Gertz. Dark patterns in the interaction with cookie banners. *arXiv preprint arXiv:2103.14956*, 2021.

[24] Maximilian Hils, Daniel W. Woods, and Rainer Böhme. Measuring the emergence of consent management on the web. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, page 317–332, New York, NY, USA, 2020. Association for Computing Machinery.

[25] Xuehui Hu, Nishanth Sastry, and Mainack Mondal. CCCC: Corralling cookies into categories with CookieMonster. In *13th ACM Web Science Conference 2021*, WebSci '21, page 234–242, New York, NY, USA, 2021. Association for Computing Machinery.

[26] International Chamber of Commerce UK. ICC UK Cookie guide, November 2012. https://www.cookielaw.org/wp-content/uploads/2019/12/icc_uk_cookiesguide_revnov.pdf; Accessed on 2021.02.08.

[27] Rolf Bagge Janus Bager Kristensen. Consent-O-Matic, 2020. https://github.com/cavi-au/Consent-O-Matic.

[28] Carlos Jensen and Colin Potts. Privacy policies as decision-making tools: An evaluation of online privacy notices. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 471–478, 01 2004.

[29] Judgement of the Court (Grand Chamber). Case C-673/17 Planet49 GmbH v Bundesverband der Verbraucherzentralen und Verbraucherverbände – Verbraucherzentrale Bundesverband e.V. ECLI:EU:C:2019:246, 1 October 2019. http://curia.europa.eu/juris/document/document.jsf?docid=218462&doclang=EN; Last accessed on: 2021.02.06.

[30] Georgios Kampanos and Siamak F Shahandashti. Accept all: The landscape of cookie banners in Greece and the UK. In *IFIP International Conference on ICT Systems Security and Privacy Protection*, page to appear. Springer, 2021.

[31] Oksana Kulyk, Annika Hilt, Nina Gerber, and Melanie Volkamer. "This website uses cookies": Users' perceptions and reactions to the cookie disclaimer. In *European Workshop on Usable Security (EuroUSEC)*, 04 2018.

[32] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. "Tranco: A research-oriented top sites ranking hardened against manipulation". In *Proceedings of the 26th Annual Network and Distributed System Security Symposium*, NDSS 2019, February 2019.

[33] Sam Macbeth. Cliqz Autoconsent, December 2020. https://github.com/cliqz-oss/autoconsent.

[34] C. Matte, N. Bielova, and C. Santos. Do cookie banners respect my choice? Measuring legal compliance of banners from IAB Europe's Transparency and consent framework. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 791–809, 2020.

[35] Mozilla. OpenWPM – a web privacy measurement framework. https://github.com/mozilla/OpenWPM, 2020. Version used: 0.12.0.

[36] Mozilla. Enhanced tracking protection in Firefox for desktop, June 2021. https://support.mozilla.org/en-US/kb/enhanced-tracking-protection-firefox-desktop.

[37] Mozilla. Firefox 86 introduces total cookie protection, February 2021. https://blog.mozilla.org/security/2021/02/23/total-cookie-protection/.

[38] Commission nationale de l'informatique et des libertés (CNIL). Cookies: financial penalties of 60 million euros against the company GOOGLE LLC and of 40 million euros against the company GOOGLE IRELAND LIMITED, December 2020. https://bit.ly/3rFIjaF; Accessed on: 2021.02.07.

[39] Midas Nouwens, Ilaria Liccardi, Michael Veale, David Karger, and Lalana Kagal. Dark patterns after the GDPR: Scraping consent pop-ups and demonstrating their influence. *CoRR*, abs/2001.02479, 2020.

[40] OneTrust. Cookiepedia. https://cookiepedia.co.uk/. Accessed on 2021.02.08.

[41] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. Cookie synchronization: Everything you always wanted to know but were afraid to ask. *CoRR*, abs/1805.10505, 2018.

[42] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. Detecting and defending against third-party tracking on the web. In *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 155–168, San Jose, CA, April 2012. USENIX Association.

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

[44] Cristiana Santos, Nataliia Bielova, and Célestin Matte. Are cookie banners indeed compliant with the law? Deciphering EU legal requirements on consent and technical means to verify compliance of cookie banners. *CoRR*, abs/1912.07144, 2019.

[45] Robert E Schapire and Yoav Freund. Boosting: Foundations and algorithms. *Kybernetes*, 2013.

[46] D. Singer and R. Fielding. Tracking preference expression (DNT) W3C working group note. https://www.w3.org/TR/tracking-dnt/, January 2019.

[47] Than Htut Soe, Oda Elise Nordberg, Frode Guribye, and Marija Slavkovik. Circumvention by design-dark patterns in cookie consent for online news outlets. In *Proceedings of the 11th Nordic Conference on Human-Computer Interaction: Shaping Experiences, Shaping Society*, pages 1–12, 2020.

[48] Konstantinos Solomos, Panagiotis Ilia, Sotiris Ioannidis, and Nicolas Kourtellis. Clash of the trackers: Measuring the evolution of the online tracking ecosystem. *arXiv preprint arXiv:1907.12860*, 2019.

[49] CookieBot Support. Unclassified cookies - how do I classify them manually? https://web.archive.org/web/20201111204915/https://support.cookiebot.com/hc/en-us/articles/360003735214-Unclassified-cookies-how-do-I-classify-them-manually-, May 2018. Accessed on: 2021.02.08.

[50] Martino Trevisan, Stefano Traverso, Eleonora Bassi, and Marco Mellia. 4 years of EU cookie law: Results and lessons learned. *Proceedings on Privacy Enhancing Technologies*, 2019:126–145, 04 2019.

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

[52] Christine Utz, Martin Degeling, Sascha Fahl, Florian Schaub, and Thorsten Holz. (Un)informed consent: Studying GDPR consent notices in the field. *CoRR*, abs/1909.02638, 2019.

[53] Daniel W Woods and Rainer Böhme. The commodification of consent. In *20th Annual Workshop on the Economics of Information Security, WEIS*, page 25, 2020.

[54] Abraham J Wyner, Matthew Olson, Justin Bleich, and David Mease. Explaining the success of adaboost and random forests as interpolating classifiers. *The Journal of Machine Learning Research*, 18(1):1558–1590, 2017.

A CMP data versus Cookiepedia

A.1 Rationale for Cookie Scraping

We considered collecting the ground truth for the training dataset by querying Cookiepedia, but we decided for scraping CMPs instead for multiple reasons, which we list below.

- The CMP descriptions are a primary source with the purpose either assigned or confirmed by the website administrator. Cookiepedia is a third-party, and despite the purposes being assigned by experts, they do not have complete information about the intentions the web administrators had.
- Scraping CMPs also allows us to analyze their compliance, which motivates client-side cookie policy enforcement.

Table 5: Performance of XGBoost when applied on our reduced cookie dataset labeled using Cookiepedia.

XGBoost	Necessary	Functional	Analytics	Advertising
F1 score	86.2% ±1.1%	59.3% ±4.7%	95.2% ±1.2%	89.0% ±1.1%
Micro F1 (accuracy): 89.2% ± 1.3%				

- Cookiepedia identifies cookies by their name and not by the more specific identifier of the name and domain. This means that cookies of the same name used by different domains for different purposes would cause noise for training.
- For a long period during the course of our study, Cookiepedia was not accessible, and as such, it would have been a single point of failure for our data collection. Individual sites with CMPs can also be inaccessible, but their distributed nature ensures that we can always collect sufficient dataset for training.

A.2 Classification using Cookiepedia labels

To better compare our approach with the work of Hu et al. from [25], we applied a sequence of transformations to bring our model assumptions closer to theirs. Namely, we applied the following changes:

1. We replace the ground truth labels of our cookie dataset with labels queried from Cookiepedia. We discard all cookies for which Cookiepedia does not have a category, thus reducing the size of our dataset by 21%.
2. We reduce the number of our training samples further by randomly sampling a single cookie for each unique name. This is necessary because Cookiepedia always assigns the same label to the same name, while our dataset from CMPs could contain cookies of the same name with different purposes. Having many duplicate names would falsify the validation score.
3. We train an XGBoost model on the new dataset, and report the per-class F1 score, and overall micro F1 score.

The resulting values are presented in Table 5. Notice that our micro F1 score, which in this setting is equivalent to the accuracy, is increased from 87.2% to 89.2%. Furthermore, this F1 score is better than the F1 score of 86.7% from [25], which can be recomputed from the reported confusion matrix, but lower than their stated micro F1 score of 94.6%.

Table 6: Set of hyperparameters used for training the model with XGBoost, listed here for reproducibility.

Parameter name	**Value**
Booster type	`'gbtree'`
Tree method	`'hist'`
Learning objective	`'multi:softprob'`,
Evaluation metrics	`'merror'` and `'mlogloss'`,
Learning rate	0.25
Maximum tree depth	32
Minimum split loss	1
Minimum child weight	3
Maximum delta step	0 (no limit)
Subsample ratio	1.0
Alpha (L1 regularizer)	2
Lambda (L2 regularizer)	1
Tree growth policy	`'depth-wise'`
Maximum bins	256

Table 7: Per-difference features overview: All features that are extracted as comparisons between two contiguous updates, sorted by timestamp.

Feature Name	**Description**
Expiry difference (1)	Expiration time difference in seconds between two updates.
"Difflib" similarity (1)	Similarity ratio between cookies, as measured by "difflib".
Levenshtein distance (1)	Levenshtein distance between two cookie updates.

B Feature extraction and hyperparameters

In the following, we provide more details about the feature extraction and the model's hyperparameters. For the most detailed overview, please refer to the extended report [3] and project documentation.[5]

Feature types. We extract three major types of features from the cookies. First, from each cookie, we extract features from its attributes, presented in Table 8. Second, with each cookie update, we store the updated features listed in Table 9. The number of updates used for the feature extraction is configurable. By default we use two, so that the classification does not require longer observations of the cookie, which is a trade-off for model performance. Finally, starting with the first update, we compute the difference to the previous version of the cookie, which are the per-difference features we show in Table 7.

Classifier hyperparameters. In Table 6 we show the parameters we selected for training the XGBoost model. They were selected through the use of a randomized grid-search and 5-fold cross-validation.

[5]The feature documentation and classifier are available at: https://github.com/dibollinger/CookieBlock-Consent-Classifier.

Table 8: Per-cookie features overview: All features that are extracted once per unique cookie. Entries marked with a * may cause issues when used within the context of a browser extension. In the parentheses after the name, we show the number of vector entries the feature takes.

Feature name	**Description**
Top names (500)	One-hot vector of the most common cookie names.
Top domains (500)	One-hot vector of the most common domains.
Pattern names (50)	One-hot vector of the most common name patterns.
Name tokens (500)	Binary indicator of English tokens in the name.
IAB vendor (1)	Binary indicator, true if domain is an IAB vendor.
Domain period (1)	Indicates whether the domain starts with a period char.
Third-party* (1)	Whether the cookie originates from a third-party.
Non-root path (1)	Whether the cookie path is not the root path.
Update count* (1)	Total number of updates encountered for this cookie.
Host-only flag (1)	Whether the "host-only" flag is set.
HTTP-only changed (1)	Whether the "HTTP-only" flag changed in any update.
"Secure" changed (1)	Whether the "secure" flag changed in any update.
"Same-Site" changed (1)	Whether the "same-site" flag changed in any update.
"Session" changed (1)	Whether the "session" flag changed in any update.
"Expiry" changed (1)	Whether the expiry changed by 1+ days between updates.
Content changed (1)	Whether the cookie content changed between updates.
Levenshtein total (2)	Mean and StdDev of Levenshtein dist. between updates.
Difflib total (2)	Mean and StdDev of Difflib similarity between updates.
Length total (2)	Mean and StdDev of the cookie value length in bytes.
Compressed total (2)	Mean and StdDev of the compressed cookie value length.
Entropy total (2)	Mean and StdDev of the Shannon Entropy of values.

Table 9: Per-update feature overview: All features that are extracted once per cookie update. The number of updates used for extraction can be specified separately.

Feature Name	**Description**
"HTTP-only" flag (1)	Binary indicator of whether the "http-only" flag is set.
"Secure" flag (1)	Binary indicator of whether the "secure" flag is set.
"Session" flag (1)	Whether the cookie is a session cookie or not.
"Same-Site" flag (3)	Whether "None", "Lax" or "Strict" is set.
Expiration time (1)	Ordinal feature, contains the expiry in seconds.
Expiration intervals (8)	Interval checks on expiry, e.g., $>$ 1 day, $<$ 1 week.
Content length (1)	Total size of the cookie's value in bytes.
Compressed length (1)	Size of the cookie value after *zlib* compression.
Compression rate (1)	Reduction of the size after *zlib* compression.
Shannon entropy (1)	Shannon entropy of the cookie update's value.
URL encoding (1)	Indicates whether the cookie value is URL encoded.
Base64 encoding (1)	Indicates that the value is potentially Base64 encoded.
Delimiter separation (9)	Delimiter (CSV) separation type and #separators.
Contains JSON (1)	Whether the value contains a JSON object.
Content terms (50)	Binary indicator of English tokens in the value.
CSV contents (5)	Try to split as CSV and detect value types within.
JS contents (11)	Try to split as JSON and detect value types within.
Numerical content (1)	Whether the value consists entirely of digits.
Hexadecimal content (1)	Whether the value represents a hexadecimal number.
Alphabetical content (1)	Whether the value is entirely alphabetical.
Identifier content (1)	Whether the value is a valid code identifier.
All uppercase (1)	Whether the cookie value has all uppercase letters.
All lowercase (1)	Whether the cookie value has all lowercase letters.
Empty content (1)	Whether the value of the cookie is empty.
Boolean content (1)	Whether the cookie value is a boolean of some form.
Locale content (1)	Whether the value includes a country identifier.
Timestamp content (1)	Whether a UNIX timestamp is in the cookie value.
Date content (1)	Whether the value contains a date term or identifier.
URL content (1)	Whether the value contains a URL of some form.
UUID content (6)	Which UUID variant, if present in the value.

C Artifact Appendix

C.1 Abstract

Our work in this paper consists of four separate components which perform different functions but depend on each others outputs. These components are as follows:

1. **The cookie consent web crawlers.** The web crawler component uses a series of Python scripts and the OpenWPM framework gather browser cookies and associated purpose categories from websites. The output of this component is a dataset of browser cookies including category labels.
2. **The feature extraction and XGBoost classifier.** This component uses the collected dataset of cookies and transforms it into a sparse matrix representation, using all properties of a browser cookie in the process. This sparse matrix, combined with the category labels, is then used to train a decision tree model using the XGBoost algorithm. This allows us to predict purpose categories for previously unseen cookies.
3. **The GDPR violation detection scripts.** Using knowledge of the articles of the GDPR, and the cookie dataset collected by the consent web crawler, these scripts identify potential GDPR violations on websites in the wild. The output of this component is a dataset of statistics detailing the prevalence of potential GDPR violations, based on 8 different methods of analysis.
4. **The "CookieBlock" browser extension.** This addon provides a privacy protection mechanism for users which automatically deletes cookies that they did not consent to. The extension uses the classifier as the central engine to decide which cookie belongs to what category. It supports Chromium-based browsers as well as Firefox.

The components have been constructed using Python 3 and JavaScript. The webcrawler in particular is based on the OpenWPM framework version 0.12.0, and requires a Linux installation. For this reason, we provide an Ubuntu VM that comes with all dependencies preinstalled. We also provide a precomputed dataset of statistics and metrics from our previous execution of these components. This includes the candidate domains used for the web crawl, the complete set of performance metrics for the XGBoost classifier and the Cookiepedia baseline, as well as all statistics and data on the GDPR Violation Detection.

No specialized hardware is required to reproduce the results of the paper, but it requires at least 8GB of RAM, and 40 GB of disk space. Due to the nature of the dataset collection, results may differ significantly if reproduced at a later date. Instructions on how to compare and validate the results are provided in the form of a detailed “README” document, containing a step-by-step guide detailing each part of the process. This is intended to help the reader better understand the results presented in the paper.

C.2 Artifact check-list (meta-information)

- **Data set:** Yes, included. The data set and VM are found at: https://doi.org/10.5281/zenodo.5568491
- **Run-time environment:** The OpenWPM crawler only runs on Linux. The other scripts and the browser extension work on Windows and Linux. An Ubuntu VM image is included.
- **Hardware:** At least 8GB of RAM needed, and approximately 40 GB of disk space. Additional CPU cores can speed up the computation, but it works with one core.
- **Run-time state:** The results are dependent on the website content, as well as the CMP implementations, which may change over time, and are out of our control.
- **Execution:** With the complete input dataset, the web crawls alone may take between 1 and 2 weeks to complete. With a reduced dataset, the full process takes a few hours.
- **Metrics:** Accuracy, precision, recall, macro-precision, macro-recall and F1 score.
- **Output:** Printed to the console, stored in SQLite databases, JSON and log files. Expected results are included for each step of the process.
- **Experiments:** Collection of the browser cookie dataset, the training and evaluation of the classifier, the GDPR Violation detection and the generation of the extension’s classifier model can all be replicated using commands manually input by the user. We provide a detailed step-by-step guide on the process.
- **How much disk space required (approximately)?:** At least 40 GB is required for the VM. While the included datasets are much smaller than this, the data that is collected and generated may quickly take up disk space.
- **How much time is needed to prepare workflow (approximately)?:** When installing the VM image, only a few minutes. When setting the scripts up natively, at most an hour.
- **How much time is needed to complete experiments (approximately)?:** A few hours.
- **Publicly available?:** Yes, all components are publicly available on Github. Links are provided in the step-by-step guide.
- **Code licenses (if publicly available)?:** The OpenWPM crawler is GPL3 licensed. Other components are MIT licensed.
- **Data licenses:** CC by 4.0 International
- **Archived:** Yes, available at: https://doi.org/10.5281/zenodo.5568491

C.3 Description

C.3.1 How to access

The artifact is publicly available and can be downloaded as a self-contained package from:

https://doi.org/10.5281/zenodo.5568491

It includes a VM image that has all components preinstalled, as well as a README that guides the user to replicate and reproduce the results. The document also contains links to the original repositories, should the user intend to install the scripts natively.

C.3.2 Hardware dependencies

The artifact requires no specialized hardware to run. A single core machine with 8GB of RAM and mor than 40 GB of disk space should be enough. The VM requires considerable size when set up, which is due to the libraries that are used, and because of the data collection that needs to be performed to replicate the results.

C.3.3 Software dependencies

If the VM image is used, only a virtualization product such as VirtualBox or VMWare is required. All other components should be ready to use. For native installations, some Python and Node libraries are required. The exact details are provided within the step-by-step guide included as part of the artifact.

C.4 Installation

The recommended method of setting up the artifact is to load the virtual machine image using VirtualBox. All further steps are documented in great detail within the README file of the artifact. In the interest of space, we will not repeat the steps here, and instead refer to the README.

C.5 Evaluation and expected results

First, we crawled 6M domains from a Tranco list collected on May 5th. Out of these, 30k were found to have the selected CMPs on them. From these websites, we collected a ground truth of 304k cookies with labels, which we used to train an XGBoost model with 84.4% weighted accuracy. In an analysis of the 30k websites, we found that a vast majority, namely 94.7% of them, contain at least one potential privacy violation. All the steps to reproduce these results together with the intermediate files of our results are documented in great detail within the README file of the artifact.

Note that the changes to websites content cause variance in the results. We try to document this variance below:

1. **Variance for the cookie consent web crawlers.** Within the large Tranco list, the number of websites with CMPs remains roughly the same over time. Among the more popular sites, the percentage of websites using the selected CMPs is higher, allowing the use of smaller input files. In the paper, we observed suitable CMPs on 0.63% of the Tranco 6M list (see Sections 2.1 and 2.2 of the paper). In the Master Thesis report, it was 1.6% for Tranco 1M Worldwide or 1.25% for Tranco Europe, and BuiltWith website reports the selected CMPs in over 3% of the top 1M websites. We observed on average 22 cookies with label per website, which depends strongly on the number of sub-pages visited for each site (discussed in the par. 3 of Section 2.3. of the paper). We did not measure the variance for the settings in the crawler, but the results should be consistent as long as you run the provided crawler from within EU.
2. **Variance in the XGBoost classifier.** The feature extraction is deterministic, extracting the same features with each execution. Training the model appears to be stable, as we observe a standard deviation of 0.23% in the accuracy. The model's balanced accuracy will drop from the reported 84.4% if you use a smaller training dataset. Additional standard deviations for each metric are provided in the dataset.
3. **Variance in the GDPR violation detection scripts.** The observed violations depends on website selection, but the results between the master thesis report and the paper varied by 4% for the number of websites with at least one type of violation. For individual violations this variance can be higher.