Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Privacy Observatory: Reproducing Privacy Studies

Semester Project

Elisa Baux

June 17, 2024

Supervisors: Prof. Dr. David Basin, Dr. Karel Kubicek, Ahmed Bouhoula

Department of Computer Science, ETH Zürich

Abstract

This project examines the process of reproducing web privacy measurement studies so that they can be deployed by an orchestration framework.

By documenting the reproduction efforts of five selected studies, we gain insights into the challenges of reproduction and principles that can be followed to limit them.

Contents

Chapter 1

Introduction

In the field of experimental computer science, the reproduction of research findings is fundamental to establishing reliable scientific knowledge. This principle is particularly crucial in the context of internet privacy measurement studies, as reproducibility not only validates findings, but also enables long-term analyses. These can reveal trends in web service practices and assess the impact of regulatory changes, providing insights into the evolving landscape of internet privacy.

Typically, researchers develop a web crawler to conduct such studies. As noted by Demir et al. [1], reproducing crawling studies presents significant challenges. Small variations in experimental setup can lead to very large discrepancies in outcomes.

1.1 Background

In a prior MSc thesis, Kast [2] implemented an orchestration framework for running crawls. This framework, which we refer to as *the Privacy Observatory*, aims to facilitate the orchestration and post-processing of privacy measurement studies in the long term. The Privacy observatory is described more in depth in section 2.1. Kast also proposed six principles (P1-P6) to facilitate long-term reproducibility, summarized in Table A.2

Kast's work builds on the theoretical foundation laid by Demir et al. [1], who explored the requirements for reproducibility in web measurement studies by surveying 117 recent research papers to derive best practices. They specified 18 criteria necessary for ensuring good reproducibility, categorized into several groups: dataset specification (C1 - C4), applications and programs used for running the crawler (C10), specific crawling environment (C11 - C14), and post-processing evaluation (C15 - C18). The criteria can be found in Table A.1 .

In this project, we assessed five publications with regards to the criteria and principles mentioned above. We then attempted to reproduce these studies, documenting the process and which criteria arose in practice.

1.2 Selected publications

The studies to be reproduced were selected from highly ranked conferences and chosen to cover a range of themes, excluding cookie-related topics, as a study on that subject is already reproduced in the Privacy Observatory. The selected studies are described, along with the methodology to reproduce them, in section 2.4.

The assessment of these studies under the criteria of Demir et al. can be found in Tables A.3 to A.4.

Chapter 2

Methodology

In the following sections, we describe the Privacy Observatory and its use, give a methodology to reproduce a study with it, as well as share the main observations from the studies selected to reproduce.

2.1 The Privacy Observatory

The Privacy Observatory is an orchestration framework designed for running crawls to reproduce web privacy measurement studies. Implemented by Patrice Kast in a prior MSc thesis, it manages the regular initiation of jobs, collection of results, and their visualization. Below is an overview of its components and usage. For more details, refer to Patrice Kast's report [2].

2.1.1 Components

PostgreSQL Database: Stores the results of studies, including overall statistics and domain-specific measurements.

RESTful API Engine: Handles the backend logic and communicates with the database.

Front-End JavaScript WebApp: Allows users to configure new studies, monitor the platform, and analyze results.

Worker: Executes the studies and reports results back to the API.

The worker executes studies defined using Docker images, with a `docker-compose.yaml` file specifying the configuration and environment variables. Containerizing these studies mitigates issues with browser binaries and dependencies, and ensures that the execution of studies is fully automated. However, this complicates progress monitoring and manual interventions.

2.1.2 Input and Output Handling

The worker starts the study by providing it with an input through a dynamically generated `input.txt` file or a predefined domain list, which can be added using the Front-End JavaScript WebApp.

For each study, we must define a `_manager.py` script to set up the environment, perform the crawl on the websites in `input.txt`, and process the results, outputting them to `output.txt`. The study shares the results formatted as JSON objects, containing both aggregated statistics ("stats") and individual domain measurements ("doms"), to the worker using this file.

```
{
"stats": {
    "successfully_loaded": 1.0,
    "with_dialog": 0.50,
        *continue with other general statistics*
    "OnlyOptIn": 0,
        *continue with statistics on results*
    },
    "doms": {
    "linkedin.com": {
            "OnlyOptIn": 0,
                *continue with specific results*
            "result" :"Dialog found"
            },
    "wikipedia.org": {"result" :"Error: no dialog
       found"}
    }
}
```

Listing 2.1: output.txt example (DarkDialogs)

2.1.3 Scheduling Studies

Studies are scheduled using crontab-style syntax, ensuring that measurements are taken consistently. The crontab command can be used to schedule tasks to run periodically at fixed times, intervals or dates with:

```
X X X X X -> command to execute
| | | | |
| | | | ----- Day of week (0 - 7) (Sunday is both 0 and 7)
| | | ------- Month (1 - 12)
| | --------- Day of month (1 - 31)
| ----------- Hour (0 - 23)
------------- Minute (0 - 59)
```

Once a study is added, it can be monitored using the Front-End WebApp under `Runs`, where users can view the status of workers and examine the results.

2.2 Artifacts

The artifact of a publication constitutes of all resources needed to reproduce a study. It typically consists of instructions, installation/setup scripts, required data, the crawler itself, and post-processing scripts.

Instructions Clear and comprehensive instructions, typically in the form of a `ReadMe` file, are essential for understanding and reproducing a study. A description of the code and architecture helps in understanding expected behavior and simplifies troubleshooting. Detailed run instructions are particularly valuable, not only to ensure consistency, but also for ease of use, as it can be hard to distinguish between components required specifically for the study and those inherited from previous work.

Setup For the successful setup and operation of a study (especially within a Docker container), it is essential to specify both the installation requirements and the environment settings. This ensures all dependencies are met. Often, studies rely on specific versions of software,which must be clearly stated to guarantee compatibility and functionality. Typically these specifications are detailed in documents such as `requirements.txt` and `environment.yaml`, or described in the `ReadMe` file.

Dataset The original dataset provides the specific websites on which the initial study was conducted. Using the same dataset can help compare consistency with original findings. While it is useful for this dataset to include results for more precise analysis, it is not imperative as the original results are published in the paper. Alternatively, the dataset can be replaced with a current list of top websites, to provide a more modern context for the findings, as we are interested in new measurements. Ruth et al. [3] have shown that studies can often use website lists that are not representative of the actual internet. Therefore, in some cases, an updated dataset can provide a more relevant context for findings.

Crawler The source code of the crawler is crucial for reproduction. Many studies use customized crawling technology, and having access to their code prevents the errors and inefficiencies that could arise from attempting to reverse-engineer the methodology presented.

Post-processing script Once the data from the crawl is collected, the post-processing script processes it to produce the study's final results. Access to the original script ensures that the data interpretation is aligned with the one from the original study.

2.3 How to Reproduce a Study

Since every study is implemented differently, the procedure to reproduce them will vary. Each will require different amounts of time and focus on certain aspects. However, every reproduction involves several key steps and considerations which we outline below.

2.3.1 Collection of artifact

The process starts with the collection of the artifact. In case it is incomplete, contacting the authors will be necessary to acquire the missing materials.

2.3.2 Implementing the Study in Docker

Reproducing a study for the privacy observatory implies containerizing it with Docker, which involves specific considerations dependant on the crawler technology specifics of the study code. Here we outline some scenarios that are likely to occur.

When using a binary for a browser, which is recommended to enhance reliability [2], it is important to ensure that all necessary dependencies are installed. Unfortunately, sometimes the artifact authors do not include dependencies, as they maintain them locally.

To address potential issues with manual time zone configuration, setting the time zone directly in the Dockerfile (e.g., ENV TZ=timezone) is advised. Furthermore, to avoid manual input requirements, set `'DEBIAN_FRONTEND'` to `'noninteractive'`. Considerations such as GPU feature utilization and shared memory usage are also important depending on the study's demands.

Docker Compose Setup Below is a template for the Docker Compose file. It is important to assign a specific tag to each image, as the Privacy Observatory does not enforce checking for updates. Hence using the `latest` tag does not guarantee that the latest version is actually being used.

```
version: "3.4"
services:
  studyName:
    image: dockerhub_username/imageName:TAG_ID
    environment:
```

```
      - environment_variables=xxx
    volumes:
      - /opt/input.txt:/opt/path/to/input.txt
      - /opt/output.txt:/opt/path/to/output.txt
```

Listing 2.2: Docker Compose configuration

2.3.3 Deploying and Monitoring the Study

Upload the Docker image to Docker Hub—making it public if the study's results are to be openly shared—and add it to the Privacy Observatory using the `docker-compose.yaml` file. Utilize the observatory's scheduling tools to run the study at desired intervals and monitor its progress through the Front-End JavaScript WebApp.

2.4 Studies

This section outlines the reproduction of four selected web privacy measurement studies for the Privacy Observatory, giving an overview of the implementation process.

2.4.1 DarkDialogs

Kirkman et al.[4] developed a system called *DarkDialogs* to automatically detect ten different design techniques in cookie consent dialogs that nudge users towards making less privacy friendly decisions. These designs are known as dark patterns. The system was then deployed on a sample of 10k websites, taken from the Tranco top list [5]. The work, published at the 2023 European Symposium on Security and Privacy, provides insights into the prevalence of dark patterns and their association with various website characteristics. We refer to this work as *DarkDialogs*.

This study uses Selenium's web scraping Python library with ChromeDriver to load and interact with websites using the Chrome browser.

To adapt this study for a Docker environment, we had to incorporate an X server using Xvfb, as Docker does not natively support graphical displays. This allowed us to remove the `--headless` argument from Chrome's launch options. Additionally, we removed the `--no-sandbox` option to prevent websites from detecting and possibly flagging the crawler as a bot.

One of the challenges we faced was adapting the unspecified version of Chrome to be compatible with Selenium. Moreover, we encountered an index out of range error, which only manifested in the absence of a cookie dialog in the crawled website. This resulted in having "error" instead of "no dialog" as a result for the website. We resolved this issue by adding an additional check to ensure that a dialog was actually found before updating the database.

Another issue arose with the configuration options intended to automate the crawling process. Unexpectedly, the option to make the program fully automated set the crawler to require manual validation during the run, which inadvertently halted the automation, blocking the crawler's progress. Although the solution was straightforward once identified, diagnosing this issue was challenging due to the lack of any indication that the system was awaiting input.

Finally, the original study did not publish a post-processing script, requiring us to implement this component to match the outputs of the published paper.

2.4.2 PURL

Munir et al. developed a system to detect and sanitize tracking information embedded in decorated links on web pages. With a machine learning approach, it effectively identifies and removes tracking data from URLs, while minimizing website breakage. The system was deployed on 20k websites sampled from the Tranco top-million websites list and found that 73.02% of sites abused link decoration for tracking, often by well-known advertisers and trackers. This work, which we refer to as *PURL*, was published at the 2024 USENIX Security Symposium and highlights the extensive use of link decoration for tracking and the need for precise detection methods.

PURL involves first running a web crawler that uses OpenWPM with Firefox. Then, the data collected from the crawl is used to construct a graph representation of webpage executions, capturing elements such as HTML DOM nodes, script executions, network requests, and URL decorations. This graph enables the extraction of features indicative of tracking behavior, which are finally analyzed using a supervised machine learning classifier to distinguish between tracking and non-tracking link decorations.

The main challenge was the lack of clear instructions for initiating the crawl, and how the rest of the code fit together. This issue was further complicated by compatibility problems with the provided Firefox binary.

After contacting PURL's authors, the repository was updated and, with a few fixes, the crawler could be run. However, the researcher encountered issues integrating OpenWPM results into the PURL pipeline and hasn't had time to resolve these problems or finish updating the README. As a result, we are still awaiting his response to complete this reproduction.

2.4.3 Everybody's Looking for SSOmething

Dimova et al. [6] developed a system to evaluate the privacy implications of OAuth-based Single Sign-On (SSO) on the web. This study, which we refer to as *Looking for SSOmething* was published in the Proceedings on Privacy Enhancing Technologies in 2023. They deployed a large-scale analysis of 100k websites from the Chrome User Experience Report [7] and reveal that 18.53% of websites using OAuth request non-minimal scopes, i.e., more user data than necessary. This work highlights how websites often request more personal information than required, raising privacy concerns.

This study involves a three-step crawling process to collect data across the web. At first, the crawler searches for login buttons on the homepage of each website. Then it clicks on each potential login button and collects OAuth buttons on the subsequent login pages. Finally, the crawler clicks on each OAuth button and extracts the scope parameter from the authorization request.

The crawling steps are performed using PyChromeDriver, instrumented headlessly via the Chrome DevTools Protocol. We faced challenges running the study in a sandboxed environment, ultimately using the `--no-sandbox` flag to resolve the issues. As mentioned earlier, this is not an optimal solution, as it causes browser fingerprinting that is used in bot detection. Additionally, the provided code was missing specific requirements and version details.

To manage the multi-step process of this study, the researchers used their laboratory's platform "dnetcrawl", which orchestrates and executes large-scale web crawls. Since this platform is restricted to their laboratory, we had to manually implement the communication between the scripts, saving the results of each step and reformatting them for the next. As our implementation does not include parallelization, the reproduction of this study is significantly slower than the original, which ran on ten virtual machines.

The shared scripts would get indefinitely stuck if a page did not load, so we integrated a timeout mechanism to ensure progression. At first we tried a retry mechanism, but it was ineffective and thus removed.

The post-processing script was initially not shared and was only obtained after communication with the researchers. Due to their busy schedules, the script was not cleaned up and was designed for use with remote MongoDB [8]. We modified it to work with local SQLite3 [9] so that it was compatible with our setup.

2.4.4 Targeted and Troublesome

Moti et al. [10] conducted an analysis of tracking and advertising practices on children's websites. Their study involved a large-scale crawl of 2,000 child-directed websites which they compiled using an ML classifier, identifying the presence of trackers, fingerprinting scripts, and targeted advertisements. They found that around 90% of child-directed websites embed one or more trackers, and about 27% contain targeted advertisements. This research, presented at the 2024 IEEE Security and Privacy, reveals the widespread occurrence of privacy violations and inappropriate ad content on websites aimed at children. We refer to this work as *Targeted and Troublesome*.

The implementation of this study involves multiple parts. At first, researchers compiled a list of 2K child-directed websites by training a text based classifier to detect children's website using HTML metadata fields.

Next, they extended the Tracker Radar Collector (TRC) [11], a Puppeteer-based [12] web crawler, to go through the compiled list of child-directed websites.

The study analysed three sets of results: the extent to which ads appearing on children's websites are targeted; an analysis of adverts from categories regarded as problematic for children; and the prevalence of online tracking, examining trackers, cookies, and the use of browser fingerprinting techniques.

Given our focus on privacy and the complexity involved in analyzing problematic ad content, we decided to concentrate on reproducing the results related to ad targeting and online tracking. This decision excludes the exploratory analysis of problematic ad content, thereby reproducing about two-thirds of the original study's results.

The shared code provides scripts for reproducing the crawls on the generated children's website lists, making it straightforward to reproduce that part. Although the unit tests from the TRC code did not pass, they were useful for debugging purposes. The changes necessary involved adapting the node version and fixing package dependency errors.

Contacting the authors was necessary to receive the post-processing code to gain insights into the data. The authors are still working on providing this code, so the reproduction of this study is not yet finished.

2.4.5 Investigating Persistent PII Leakage-Based Web Tracking

Dao et al. [13] developed a system to investigate persistent personally identifiable information (PII) leakage-based web tracking. Their study, which we refer to as *Persistent PII*, analyzes how PII is leaked during the authentication flows of 307 popular shopping sites from the Tranco top 10k list. The researchers found that 42.3% of these sites leak PII to third-party services.

For this study, researchers manually signed up to websites using fabricated personas to avoid biases introduced by automated bots. This method allowed them to analyze how personally identifiable information (PII) is handled during the sign-up process by collecting HTTP requests, responses, and cookies.

Due to the manual nature of this data collection and the specific methodologies involved, reproducing this study within the Privacy Observatory was infeasible.

Chapter 3

Results

In this chapter, we present the outcomes of our attempts to reproduce the selected privacy studies, providing a general overview and exploratory results from the two completed studies.

3.1 General results

3.1.1 Overview of reproduction attempts

Table 3.1: Overview of our attempts at reproduction of selected studies, with time taken, the resources used, the status (S = successful, U = unfinished, F = failed due to inefficiency) and the number of messages exchanged with original study author.

Publication	Time	Resources reused	Status	Messages
DarkDialogs	25h	Dataset, Crawler	S	0
PURL	22h	Crawler	U	8
Targeted & Trouble.	14h	Dataset, Crawler	U	8
Looking for SSOme-thing	47h	Minimal crawler, Post-processing script	F	8

Two studies; *PURL* and *Targeted and Troublesome* remain unfinished, as we are waiting for necessary resources from their respective authors. As mentioned earlier, *PURL* requires fixes to its pipeline and code documentation. We are waiting to receive post-processing scripts needed to analyze the data from the crawls of *Targeted and Troublesome*.

3.2 Reproduced studies

3.2.1 DarkDialogs

We conducted five crawls on a set of 11 websites to test our reproduction of the DarkDialogs study. The precise results are detailed in A.7.

On average, each crawl took approximately 86 seconds per website. Extrapolating this to the original dataset of 11,000 websites, suggests it would take approximately 11 days of non-stop operation to complete the study. This is a preliminary estimate and assumes linear scalability, which may not hold due to other operational factors. Nevertheless, it provides a rough indication that the study is feasible to run on a large scale.

The first two runs were spaced seven days apart, followed by two additional runs with a two-day interval, and the final two runs were performed on the same day. The results remained identical across all runs, with the only variation observed being the time taken for each crawl. This consistency in the results demonstrates their reliability.

We attempted running a crawl on a list of 500 websites, but we encountered an issue in the part of the code that processes text found on clickable elements. This error comes from the Google Translate Ajex API failing to respond to our requests, probably having blocked our server.

3.2.2 Looking for SSOmething

To test the reproduction of "Looking for SSOmething", we ran the crawler five times (SSO_1 to SSO_3 in Table A.8). We analyzed five websites, among which only `wikipedia.org` does not have an OAuth button. We compared our findings of the OAuth scopes with those reported in the original study, as shown in Table A.9.

On average, each website took over 30 minutes to process. By contrast, the original study completed its analysis across 100,000 websites from the Chrome User Experience Report within 20 days. Given our pace, a similar scale would require more than 2,000 days, rendering this replication impractical.

Concerning the results:

- The site `bookmeter.com` consistently failed to load during our trials.
- When manually examining the raw results of the four sites from the original study, we observed that, with the exception of `spotify.com`, their results are accurate. The error is a missing `apple` button, likely added after the original study was conducted.
- In every trial, none of `connectparts.com.br`'s OAuth buttons were detected.

- In one out of four instances, `pakwheels.com`'s buttons were not detected. The issue occurred during the last step, which visits OAuth elements, although the correct buttons were identified by the previous scripts, this one did not recognize them as OAuth buttons.

From these results, we can conclude that our reproduction is too unstable and inefficient to be usable.

Chapter 4

Discussion

In this chapter we discuss the results from the two reproduced studies, overview the categorization of issues encountered, and propose three additional reimplementation principles.

4.1 Reproduction results

From the results of our test runs, two main questions arose. In this section we provide possible answers.

4.1.1 Consistency

Why is Looking for SSOmething so inconsistent compared to DarkDialogs?

Compared to *DarkDialogs*, which loads each website once to perform the crawl, Looking for *SSomething* reloads a website at each step, for every result it retrieved in the previous one. This can lead to more inconsistencies, due to several factors :

Content variability: The website content can change between crawls in separate steps. This can cause the script to miss elements altogether or interact with different elements in subsequent runs.

Dynamic content: JavaScript-driven dynamic content, including OAuth buttons that appear conditionally or change properties, could lead to different interactions each time the script runs.

Crawler detection: The crawler could be getting muted or blocked as a website might recognise subsequent connection, particularly in this case because it is run with the `--no-sandbox` option.

4.1.2 Performance

Why are the results of Looking for SSOmething" significantly worse than the original study?

The discrepancy in the results is likely linked to our lack of access to the complete codebase of the original crawler. Consequently, we had to implement the management of the data and error handling ourselves. Writing a substantial amount of code independently could have introduced errors and bugs as some assumptions made during the implementation might have been incorrect.

Moreover, while running our reproduction attempt, we encountered many errors associated with the `pychrome` package. Although we were told by the author that these errors can be ignored, it could be possible that they introduced instability and affect our results. Additionally, the consistent occurrence of these errors could obscure other significant errors that should not be overlooked, further impacting our results.

4.2 Categorization of issues

4.2.1 Encountered issues

In this section, we detail the issues encountered and how many would have been avoidable by following principles by Kast.

P2: Limit external dependencies In *DarkDialogs*, the use of the Google Translate Ajax API led to an error while running a crawl on the Tranco top 500 websites, likely due to our server being banned from the service.

P3: UnstabManualle browser binaries When reproducing *Darkdialogs* a significant amount of time was spent finding the right version of Chrome, which was not available online. Similarly, *PURL* had a browser binary available, but it was non-functional, so it is imperative that they are stable.

P6: Reimplementation guidance Considerable time was required to understand how to run *PURL*. Providing clear instructions and outlining expected behavior is essential to enhance reproducibility.

C9: Used crawler is publicly available Both *PURL* and *Looking for SSOmehing* satisfy this criterion, but Demir et Al's criteria is insufficient when it comes to artifacts. While the focus is on sharing code and implementation details, there is a discrepancy between what is documented in papers and the usability of the published code. This gap should be addressed to improve reproducibility.

4.2.2 Non categorized issues

From our observations we identified three additional principles that should be taken into account for successful reproduction.

P7: Error handling Code which causes errors can distract attention and lead to reproducibility issues. It is important to document any normal errors so that efforts are not wasted trying to fix non-issues.

P8: Efficiency Efficiency is crucial, as demonstrated by *Looking for SSOmething*. Shared code should be efficient to run and not excessively time-consuming. Efficient crawling facilitates testing the code on subsets, allowing quicker identification and correction of issues (e.g., errors in output format).

P9: Manual effort necessary Automated runs are essential for reproducibility. Studies requiring manual work, such as the *Persistent PII* study, cannot be reproduced effectively in automated environments.

Chapter 5

Conclusion

This project has highlighted the complexity of creating a generalized base for reproduction of privacy measurement studies, as each study is implemented differently and will present its own unique challenges. We reproduced two studies with the Privacy Observatory, one reproduction was successful, while the other was inefficient and unstable.

From our attempts, we were able to build upon Kast's findings to identify additional principles that, if followed, could significantly ease the process of reproduction. Moreover, we observed a discrepancy between what is documented in published studies' papers and the actual artifacts shared, revealing a weakness in assessing the reproducibility of a study based solely on its written description.

Bibliography

[1] N. Demir *et al.*, "Reproducibility and replicability of web measurement studies," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 533–544.

[2] P. Kast, "Privacy observatory aggregation system for reproduction of privacy studies," M.S. thesis, ETH Zurich, 2023.

[3] K. Ruth, D. Kumar, B. Wang, L. Valenta, and Z. Durumeric, "Toppling top lists: Evaluating the accuracy of popular website lists," in *Proceedings of the 22nd ACM Internet Measurement Conference*, ser. IMC '22, New York, NY, USA: Association for Computing Machinery, 2022, pp. 374–387. DOI: 10.1145/3517745.3561444.

[4] D. Kirkman, K. Vaniea, and D. W. Woods, "Darkdialogs: Automated detection of 10 dark patterns on cookie dialogs," in *2023 IEEE 8th European Symposium on Security and Privacy (EuroSP)*, 2023, pp. 847–867. DOI: 10.1109/EuroSP57164.2023.00055.

[5] V. L. Pochat, T. V. Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, *Tranco: A research-oriented top sites ranking hardened against manipulation*, 2021. [Online]. Available: https://tranco-list.eu/.

[6] Y. Dimova, T. van Goethem, and W. Joosen, "Everybody's looking for ssomething: A large-scale evaluation on the privacy of oauth authentication on the web," *Proc. Priv. Enhancing Technol.*, vol. 2023, pp. 452–467, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259940376.

[7] *Chrome ux report*, Google Chrome, 2022.

[8] *Mongodb: The developer data platform*, [Accessed 10 Jun. 2024], MongoDB, Inc., 2023. [Online]. Available: https://www.mongodb.com.

[9] R. D. Hipp, *SQLite*, version 3.31.1, 2020. [Online]. Available: https://www.sqlite.org/index.html.

[10] Z. Moti *et al.*, "Targeted and troublesome: Tracking and advertising on children's websites," *ArXiv*, vol. abs/2308.04887, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:260735831.

[11] *Tracker radar collector*, https://github.com/duckduckgo/tracker-radar-collector, 2023.

[12] *Puppeteer*, https://github.com/puppeteer/puppeteer, 2023.

[13] H. Dao and K. Fukuda, "Alternative to third-party cookies: Investigating persistent pii leakage-based web tracking," in *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '21, Virtual Event, Germany: Association for Computing Machinery, 2021, pp. 223–229, ISBN: 9781450390989. DOI: 10.1145/3485983.3494860. [Online]. Available: https://doi.org/10.1145/3485983.3494860.

Appendix A

Appendix

A.1 Principles and reimplementation criteria

The principles for reproduction best practice can be found in Table A.2, Table A.1 contains the description of the criteria for reproduction defined by Demir et al. [1] and the assessment of the four selected studies is found in Tables A.3 to A.6.

A.2 Results from test crawls

The results of the test run of *DarkDialogs* can be found in A.7, whilst the results of the test runs of *Looking for SSOmething* and the Oauth buttons found are in Tables A.8 and A.9 respectively.

Table A.1: Description of the reimplementation criteria according to Demir et al.

Category	ID	Description
Dataset	C1	State analysed sites: States used dataset, toplist, or user clickstreams, including version.
	C2	State analysed pages: Offers a .csv or comparable with all analysed pages (i.e. distinct URLs).
	C3	State site or page selection: Discusses the selection process of analysed sites.
	C4	Perform multiple measurements: Discuss which pages are analysed in consecutive measurement runs, if appropriate.
Experiment Design *Building the Crawler*	C5	Name crawling tech.: Describes the used crawling technology (e.g. OpenWPM).
	C6	State adjustments to crawling tech.: States which technology features were used and/or (slightly) adjusted.
	C7	Describe extensions to crawling tech.: Describes which features were developed to conduct, if any.
	C8	State bot detection evasion approach: Discusses which means were taken that the crawler was not detected, if necessary.
	C9	Used crawler is publicly available: Provides the crawler in a public location.
	C10	Mimic user interaction: Describes how the user interaction was implemented, if applicable.
Experiment Design *Experiment Environment*	C11	Describe crawling strategy: Describes which crawling strategy was used (e.g. stateless vs. stateful).
	C12	Document a crawl's location: States from which location(s) the study was conducted.
	C13	State browser adjustments: Discusses properties of the browser (e.g. user agent, version, used extensions).
	C14	Describe data processing pipeline: Describes the data processing steps in detail.
Evaluation	C15	Make results openly available: Authors provide the (raw) measurement results.
	C16	Provide a result/success overview: Describes the outcome of the measurement process on a higher level.
	C17	Limitations: Discusses the limitations of the experiment.
	C18	Ethical discussion: Discusses ethical implications of the experiment (e.g. exploiting vulnerabilities).

Table A.2: Reproducibility principles identified by Kast

Principle	Description
P1	Dockerfile vs. Docker image
P2	External dependencies
P3	Unstable browser binaries
P4	Garbage collection
P5	Certificate expiry
P6	Reimplementation guidance

Table A.3: Demir et al. Criteria for DarkDialogs

Criterion	Status	Justification
C1	Satisfied	*Tranco top list October 2021*
C2	Satisfied	*https://doi.org/10.7488/ds/3475*
C3	Satisfied	*"A sample of 10K websites provides enough statistical power. Collecting a larger sample incurs a financial/climate cost that is arguably unnecessary."*
C4	Omitted	
C5	Satisfied	*Selenium web scraping python library*
C6	Doesn't apply	*They implemented the crawler*
C7	Doesn't apply	*They implemented the crawler*
C8	Omitted	
C9	Satisfied	*"We released the source code of the DarkDialogs system in a public repository, along with system installation and usage instructions." github.com/DarkDialogs/OpenScience*
C10	Omitted	
C11	Satisfied	*Stateful*
C12	Undocumented	*Uk-based VPN*
C13	Satisfied	*Uses ChromeDriver to load and interact with the websites using the Chrome browser*
C14	Omitted	
C15	Satisfied	*https://doi.org/10.7488/ds/3475*
C16	Satisfied	*Discusses results*
C17	Satisfied	*Discusses it, Section 6.4*
C18	Omitted	

Table A.4: Demir et al. Criteria for PURL

Criterion	Status	Justification
C1	Satisfied	*Tranco top-million websites (2019)*
C2	Satisfied	*https://github.com/purl-sanitizer/purl/blob/main/OpenWPM/sites.csv*
C3	Satisfied	*They explain their choice to ensure that their crawls cover the most popular websites as well as lower-ranked websites of varying popularity*
C4	Omitted	
C5	Satisfied	*OpenWPM (v0.17.0)*
C6	Undocumented	*States that adjustments were made but did not specify how*
C7	Satisfied	*OpenWPM extended to record execution information across HTML, network, JavaScript, and storage layers during a webpage load*
C8	Satisfied	*Uniformly at random wait an additional 5–30 seconds for bot mitigation.*
C9	Satisfied	*For reproducibility and to foster follow-up research, PURL's source code is available at https://github.com/purl-sanitizer/purl.*
C10	Satisfied	*For each site, we crawl its landing page, randomly scroll and move the cursor, and then select up to 20 internal pages to visit at random*
C11	Satisfied	*Crawler was used in a stateless environment*
C12	Undocumented	*We conduct our crawls from the vantage point of an academic institution in the US.*
C13	Satisfied	*Firefox (v102) and We turn off all built-in tracking protections provided by Firefox (Enhanced Tracking Protection [ETP])*
C14	Satisfied	*States that adjustments were made but did not specify how*
C15	Satisfied	*https://github.com/purl-sanitizer/purl/tree/main/data*
C16	Satisfied	*Analysis of PURL with ground truth*
C17	Satisfied	*Discusses it, for example Purl is not suitable for runtime deployment*
C18	Omitted	

Table A.5: Demir et al. Criteria for Looking for SSOmething

Criterion	**Status**	**Justification**
C1	Satisfied	*100k websites of the Chrome User Experience Report (July 2021)*
C2	Satisfied	*Not public yet*
C3	Satisfied	*They ensured focus on frequently visited sites by using the CrUX list of top websites*
C4	Omitted	
C5	Satisfied	*Chrome Devtools Protocol (pychrome)*
C6	Doesn't apply	*They implemented the crawler*
C7	Doesn't apply	*They implemented the crawler*
C8	Omitted	
C9	Satisfied	*The full code that we used to detect SSO buttons can be found in our public repository (however it is not yet)*
C10	Omitted	
C11	Satisfied	*Crawler was used in a stateful environment*
C12	Undocumented	*Based in the EU*
C13	Omitted	
C14	Omitted	
C15	Satisfied	*Not public yet*
C16	Satisfied	*Discussed in results*
C17	Satisfied	*Discusses it, Section 9.3*
C18	Satisfied	*Discusses it, Section 3.6*

Table A.6: Demir et al. Criteria for Targeted and Troublesome

Criterion	Status	Justification
C1	Satisfied	*State Analyzed Sites - June/July 2022 crawl archive now available – Common Crawl dataset*
C2	Undocumented	*There is a csv file in the code artifact but it is not mentioned in the paper*
C3	Satisfied	*Uses a curated lists and a classifier to generate a comprehensive list of child-directed websites*
C4	Omitted	
C5	Satisfied	*The study extends Tracker Radar Collector (TRC)*
C6	Satisfied	*They extended it with extensions*
C7	Satisfied	*New collectors, such as FingerprintCollector, LinkCollector, VideoCollector, and AdCollector, were added*
C8	Satisfied	*The study uses TRC's anti-bot measures*
C9	Omitted	
C10	Satisfied	*For mobile crawls, emulating a mobile browser including spoofing viewport dimensions, touch support, and user-agent string*
C11	Satisfied	*Stateless*
C12	Satisfied	*Frankfurt, Amsterdam, London, San Francisco, and New York City*
C13	Omitted	
C14	Satisfied	*Analyse photos with cloud vision API and we get the statistics*
C15	Satisfied	*"We are working on preparing and documenting the dataset for release." On github.com/targeted-and-troublesome*
C16	Satisfied	*The study describes the outcome of the measurement process, including the prevalence of trackers and targeted advertisements on child-directed websites*
C17	Satisfied	*Discusses the limitations of the experiment. Section 6.3*
C18	Satisfied	*The study discusses ethical considerations. Section 6.2*

Table A.7: Results from runs of DarkDialogs

Statistic	DD_1	DD_2	DD_3	DD_4	DD_5
Time Taken (seconds)	918.54	1025.93	1001.75	827.24	987.78
Successfully Loaded	1.0	1.0	1.0	1.0	1.0
With Dialog	0.545	0.545	0.545	0.545	0.545
With Cookie on Load	0.833	0.833	0.833	0.833	0.833
With ID Cookie on Load	0.667	0.667	0.667	0.667	0.667
Only Opt-In	0.0	0.0	0.0	0.0	0.0
Opt-Out More Cookies	0.182	0.182	0.182	0.182	0.182
Highlighted Opt-In	0.0	0.0	0.0	0.0	0.0
Obstructs Window	0.0	0.0	0.0	0.0	0.0
Complex Text	0.273	0.273	0.273	0.273	0.273
More Options	0.545	0.545	0.545	0.545	0.545
Ambiguous Close	0.182	0.182	0.182	0.182	0.182
Multiple Dialogs	0.0	0.0	0.0	0.0	0.0
Preference Slider	0.0	0.0	0.0	0.0	0.0
Close More Cookies	0.091	0.091	0.091	0.091	0.091

Table A.8: Results from Looking for SSOmething test runs

Metric	SSO_1	SSO_2	SSO_3	SSO_4
Total Time Taken (seconds)	10702.70	11339.66	9858.29	9016.50
Total Sites	5	5	5	5
Loaded Sites	4	4	4	4
Sites with OAuth found	2	1	2	2
Buttons found	4	2	4	4
Sites w OAuth present	4	4	4	4
Buttons present	10	10	10	10
Correctness (Sites)	50.0%	25%	50%	50%
Did Not Load	20.0%	20.0%	20.0%	20%

Table A.9: Scopes and OAUth buttons found during different test crawl.With 0 = original study and i = SSO_i

Website	Provider	Permissions	0	1	2	3	4
bookmeter.com	Google	profile	✓	×	×	×	×
		openid	✓	×	×	×	×
		email	✓	×	×	×	×
	Twitter	content_write	✓	×	×	×	×
	Facebook	public_profile	✓	×	×	×	×
		email	✓	×	×	×	×
spotify.com	Google	openid	✓	✓	✓	✓	✓
		email	✓	✓	✓	✓	✓
		profile	✓	✓	✓	✓	✓
	Facebook	default	✓	×	×	×	×
	Apple	email	×	✓	✓	✓	×
		name	×	✓	✓	✓	×
connectparts.com.br	Facebook	email	✓	×	×	×	×
	Google	email	✓	×	×	×	×
		profile	✓	×	×	×	×
pakwheels.com	Facebook	email	✓	✓	✓	✓	✓
	Google	email	✓	✓	×	✓	✓
		profile	✓	✓	×	✓	✓

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Privacy Observatory: Reproducing Privacy Studies

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	First name(s):
Baux	Elisa
	Estell

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	Signature(s)
Zurich, 17 June 2024	EBaux

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.