ExPD: Semi-automated Web Extraction of Personal Data

Alexander Böhner[1] Dominik Herrmann[2]

Abstract: According to the GDPR, data subjects can issue a Subject Access Request (SAR) to obtain all personal data concerning them from a data controller. Data controllers are interested to automate the handling of SARs, which is challenging for legacy applications and services that lack suitable export functionality. This challenge is addressed by the ExPD prototype, a tool that automates the task of collecting personal data from the pages of web applications. The ExPD operator specifies extraction rules interactively in the browser using a small set of pages. After extraction, a tool-assisted refinement stage allows to fine-tune the exported data. Besides automating the processing of SARs, ExPD may also be useful for citizens who want to obtain an overview of their public data footprint on sites that share user contributions, which is demonstrated in a case study.

Keywords: Subject Access Requests; Crawler; Case Study

1 Introduction

Article 15 of the General Data Protection Regulation (GDPR) of the European Union grants data subjects the right to obtain all personal data concerning them from a data controller. In this paper, we focus on data controllers that run web-based services and applications. This includes, for instance, Software-as-a-Service providers but also employers who provide internal web applications that are used by their employees. While large providers have started to offer self-service portals (e. g., Google Takeout), many small providers process Subject Access Requests (SARs) manually by compiling a human-readable report with data collected from various applications.

Manual collection is time-consuming and error-prone, resulting in dissatisfactory responses [AD18, HL16, KLH20, MAvE18, Ur19]. When services and legacy tools lack dedicated export functionalities, some data controllers resort to sharing screenshots with the requested information [HL16]. In at least one documented case a controller even responded with personal information of the wrong subject [KLH20].

According to a 2020 Gartner report, there is increasing interest in the automation of SARs [Ga20]. An early attempt from 2018 used Robotic Process Automation (RPA), which is a sophisticated version of recording and replaying manual interactions with software [Fi18].

[1] University of Bamberg, Privacy and Security in Information Systems Group, 96045 Bamberg, Germany alexander.boehner@uni-bamberg.de
[2] University of Bamberg, Privacy and Security in Information Systems Group, 96045 Bamberg, Germany dh.psi@uni-bamberg.de

There are, however, no public reports about the used techniques, their effectiveness, and their limitations. What is a feasible approach and what are its limitations? To address this question, we present the *ExPD* prototype, a lightweight tool for semi-automated extraction of personal data from web applications. ExPD runs within the browser and records to what parts of an application a human operator navigates and what parts of the pages the operator marks for extraction. From that, ExPD generates XPath rules and a graph model. The model is used in subsequent runs of ExPD to automatically navigate through the whole web application, extracting only the parts on the relevant pages that concern a given (parametrized) data subject (identified by, e. g., a user ID).

ExPDs targets both *data controllers* and *data subjects*: ExPD enables data controllers to extract personal data without knowledge about programming or databases. Because to this day data controllers reportedly fail to provide satisfactory responses to SARs, ExPD enables data subjects to collect at least those pieces of personal data that are exposed on the web interface of an application. As ExPD rules are typically agnostic of a particular user identity, data subjects could share ExPD rule sets for popular services among themselves.

Outline Section 2 presents related work. After summarizing the requirements in Sect. 3, we explain our approach in Sect 4. After that, we present a case study in Sect. 5 and discuss our approach in Sect. 6, before we conclude in Section 7.

2 Related Work

There are various approaches for data extraction from web pages. *Wrapper induction* systems [LRNS02, Fu14, Lo18] generate programs that extract data from web sources. These approaches are not suitable for our scenario, since they do not traverse an application but are meant to work with a predefined set of pages. The set of pages that contain personal data may change when content is added.

Record and Replay systems record browser interactions and produce scripts for replaying the recorded interaction [AC11, Ch15, CMB18, Li09, Ne15]. ExPD is similar to existing Programming by Demonstration (PBD) approaches that are designed for end-users and do not require web or programming knowledge. Like ExPD, Vegemite [Li09], Webcombine [Ch15], und Rousillon [CMB18] can *expand* the recorded interactions. Expansion improves the usability for operators because it decreases the number of interactions that have to be recorded.

Handling SARs requires the precise extraction of the personal data of a particular subject, which is not supported by the existing tools. We did not find ready-to-run code to extend one of the existing tools. Thus, we resorted to building an ExPD proof-of-concept application from scratch, allowing us to tailor it to the specific set of requirements.

3 Requirements

ExPD is supposed to meet the following three requirements:

R1: Precise Extraction ExPD is required to extract personal data of one specific data subject only – even if a page or a relevant fragment contains data of other subjects. ExPD must enable operators to precisely specify the desired data before collection as well as allow them to review and sanitize the data after extraction.

R2: Efficient Usage Specifying the desired data should be as effortless as possible and not require programming skills. Data spread over reoccurring structures or multiple pages (pagination) should be discovered and collected automatically. It should also be possible to parameterize the data subject so that a given specification can be used to handle SARs for various subjects.

R3: Simple Deployment Setup of ExPD should be easy to encourage adoption by data controllers, researchers, and data subjects. It should run as an application in a desktop environment without depending on third-party services or infrastructure.

4 Approach

ExPD uses three phases to extract personal data from web applications. Firstly, the *learning phase* builds a graph model based on the operator's input. Secondly, the *extraction phase* uses this model to automate both the interaction and extraction in the application. Thirdly, the *refinement phase* presents extracted elements to the operator, who decides what data to export from these elements. In the following, we describe each of the three phases separately. Before that, we clarify the basic concepts of how we model a web application using an example application.

4.1 Example application

We explain our approach with the following example web application, a web application where users can share comments (Fig. 1). The application provides:

1. a list of comments made by different users,
2. detailed descriptions of particular comments,
3. a more accurate description of a comment's date of origin as tooltip,
4. and a listing of the latest logins of the current user (view not included in Fig. 1).

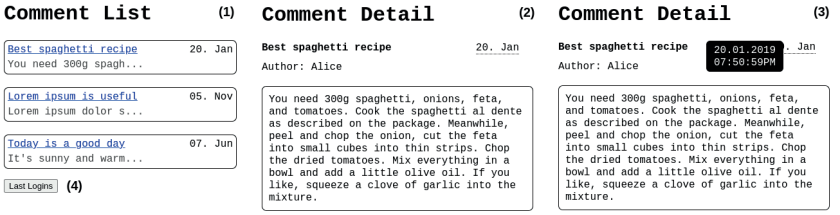

Fig. 1: Comment-sharing web application.

The application displays a comment's description after clicking on the comment in the list. In a comment's description view, hovering with the cursor above its year of origin will show the exact date of creation in a custom-styled tooltip.

In our example application, we want to extract the personal data of the data subject that is a certain user in the application. Personal data is present in (2) and (3) of the application. To screen these pieces of data, our system has to navigate from (1) over (2) to (3). Then, our system has to locate the personal data on the webpage and eventually persist it.

To explain our approach, we first define our terminology with the example application: In the application, (1), (2), and (3) each result from an interaction with the browser, namely, entering a URL, clicking a button, and hovering to reveal a tooltip. Each interaction with the browser leads to an *application state*. We define an application state as the result of an interaction with the application. For instance, a click on a comment in (1) is an interaction that results in the application state shown in (2). A similar interaction can be done with each comment in the list, each resulting in a separate application state. However, in all these states, the application describes a comment in more detail. The states, therefore, share the same concept in the domain of the application. Additionally, the states resulted from similar interactions, i. e., clicking a comment in the list. Because the states are similar regarding the interactions and their domain concept, we introduce the *statetype*. A statetype models similar states. The statetype is the central concept of modeling an application in our system. We assume that states of the same statetype provide

- similar interactions,
- similar patterns in the DOM,
- representation of the webpage, and
- similar types of data.

Furthermore, the notion of statetypes allows us to formalise the interactions between statetypes as *interaction rules*. An interaction rule consists of both an action to take and HTML elements in the webpage, located by XPath. For instance, the list of comments in (1) is formalized as an XPath expression defining the comment elements and the action "click".

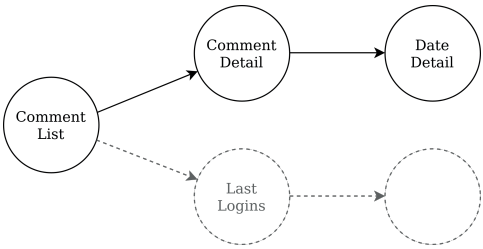

Fig. 2: Graph model of the Comment-sharing web application. Each node represents a statetype.

Clicking each of those elements results in a state of the same type as in (2). Therefore, applying an interaction rule of a statetype results in states of equal statetype.

In contrast to interaction rules, *extraction rules* define elements that contain personal data. Extraction rules use XPath to identify elements in the DOM of an application state, as well. For instance, in (2), an extraction rule would define the element containing the comment's author. The same extraction rule applies to all other application states that share the same statetype as (2) because each of them describes a comment with its author. Extraction rules additionally may specify another element that enables *conditional extractions*. Such rule only extracts if the conditional element contains given keywords that identify the data subject. For instance, in (2), an extraction rule would define the comment's full text only to be extracted if the "author" element contains the keyword "Alice".

Statetypes, interaction rules, and extraction rules are used to build a graph model of the web application. In this model, statetypes are nodes connected by interaction rules as edges. Together, they model possible interactions in the web application. For the example above, Fig. 2 illustrates a possible graph. Here, the statetypes map to the sample states in Fig. 1. At each node, extraction rules may be specified (not shown in Fig. 2).

4.2 Learning Phase

To construct the graph, ExPD captures the operator's input in an instrumented browser. At the beginning of the learning phase, ExPD opens a given URL for the desired web application. ExPD executes a JavaScript script in the instrumented browser to display a control overlay for the operator's input (Fig. 3). With this overlay, the operator may specify interaction and extraction rules for the start state in the graph model. To define a rule, the operator selects page elements by clicking the highlighted bounding rectangles. For these elements, the corresponding action can be defined in the control overlay. For instance, the operator may select multiple comment elements in the list with the action "click" to crawl as a list. The resulting interaction rule is shown in Figure 4.

In this rule, the XPath expression defines all "li" elements in the comment list. The expression captures all "li" (comment) elements, because no index for these in the correspondig DOM

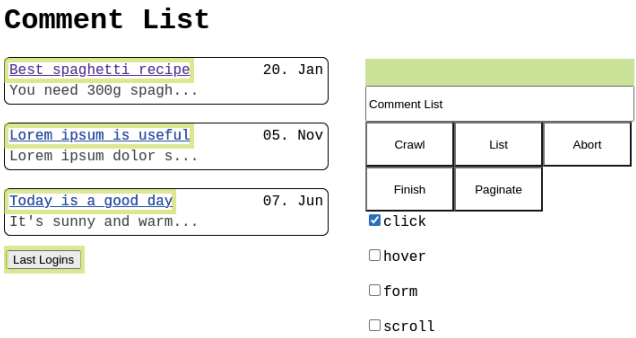

Fig. 3: User interface for operating the ExPD prototype. The control overlay and bounding boxes of elements are used for the definition of rules.

```
'action':'click'
expression':'body[1]/div[5]/ul[2]/li/div[1]'
```

Fig. 4: Interaction rule for clicking on items in a list. The “li” segment applies to all “li” elements in that level of hirarchy.

hirarchy is given. ExPD appends the rule to the start state in the graph model. After the operator has specified the rules, clicking the “Finish” button in the overlay completes the processing of the current state. ExPD then picks an interaction rule and executes it in the browser. The application reaches a new state exemplary for a different statetype than the start state. ExPD appends a new statetype in the graph with the just executed interaction rule as the edge to the start state. As with the start state, the operator is asked to specify interaction rules which eventually form an edge to a future statetype. This way, ExPD builds the graph statetype by statetype until there are no interaction rules left to execute. If an interaction rule of a different statetype than the current should be executed, ExPD first navigates to a state of the corresponding statetype. For this, ExPD calculates the shortest path in the graph model and executes the resulting interactions accordingly. To avoid navigating through the application for each new statetype, ExPD executes interaction rules in a depth-first manner.

In the process of defining rules for statetypes, the operator may also define extraction rules with the overlay controls. In contrast to interaction rules, the operator may specify a conditional element for extraction rules. The keywords used in conditional extractions are specified in a configuration file.

ExPD also supports interaction rules that are executed before reaching the start state. The rules are defined by the operator before the regular learning phase. They are called *one-time rules* because they are executed only once in the extraction phase, before the actual extraction. With these rules, the operator may log in as a specific user, click away cookie banners, or do any other task required before the extraction.

4.3 Extraction Phase

Once all rules have been added, the operator starts the extraction. ExPD uses the learned application model in the extraction phase to automatically extract personal data. First, the operator has to define an URL as the starting point in the configuration file. The application state resulting from opening the URL has to be of the same statetype as the start state in the graph. In addition to the URL, keywords for conditional extractions have to be defined in the configuration file. Because both the URL and keywords can be parametrized, the same learned model can be used for extractions with different data subjects. For instance, ExPD could start at different user profiles (one for each data subject) in the first state and perform conditional extractions only if that user's name is present.

ExPD begins the extraction at the provided URL. ExPD searches for possible interactions of each interaction rule in the corresponding statetype. For instance, all comment links in the list are possible elements to click on according to the rule in Fig. 4. ExPD appends the current application state, including the possible interactions to the corresponding statetype in the model. Analogous to the learning phase, ExPD picks an interaction to execute and explores new application states. At each state, possible interactions are determined and appended to the graph. ExPD terminates if it executed all possible interactions at least once. Because each interaction rule may result in multiple possible interactions at each state, ExPD traverses many more states than the operator used for learning the model. Still, ExPD does not leave the application model, because each visited state is linked to a statetype.

While ExPD navigates through application states, extraction rules are applied at each visited state. Extraction rules are looked up in the corresponding nodes of the statetypes. ExPD saves extracted elements that match the XPath expressions in the extraction rules at the application state in the graph.

4.4 Refinement Phase

After the extraction phase, the operator can review the extracted elements and determine the final personal data export. The operator can browse the extraction rules of each statetypes and view the extracted data, as shown in Fig. 5. Here, the operator may select text in the extracted elements to form refinement patterns. Refinement patterns specify either an HTML element's attribute or the innerText. The patterns can apply to all other data of the same rule. Therefore, the operator may specify patterns and then apply these to all extracted pieces of data to save time. Additionally, the operator can exclude elements to avoid exporting, e. g., business secrets, or other data subjects' data. Once the operator has decided on the personal data to export, ExPD prompts for additional data such as the purposes of the processing, or the categories of personal data. Finally, the operator can export the data in PDF, CSV, or JSON format.

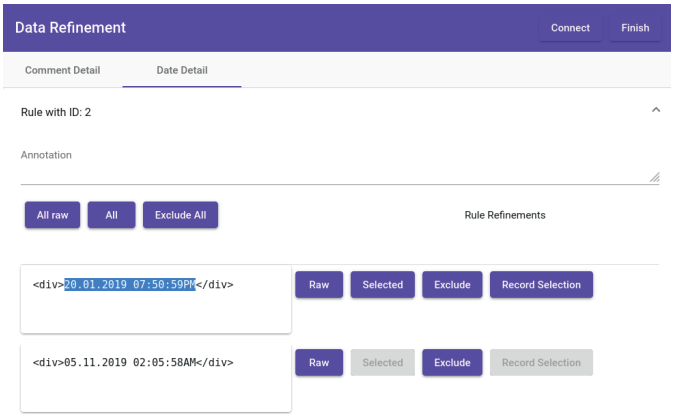

Fig. 5: Refinement interface of the prototype for selecting personal data in extracted elements.

4.5 Technical Details

The ExPD prototype[3] is implemented in two modules. ExPD's core is a Python application that instruments the browser with Selenium WebDriver[4]. We use Google Chrome as the instrumented browser. The refinement UI is developed as a seperate angular application. Both modules use the Neo4J Graph database[5] for managing the application graph model.

5 Case Study

In this section we demonstrate the ramifications of using ExPD in practice using the question and answers platform *stackoverflow.com* as an example. This platform accepts SARs via a web form. For the case study, however, we are interested in the use case of a private individual who wishes to obtain their publicly accessible data footprint. We will collect selected personal data with ExPD. To obtain a ground-truth baseline, we will use the public API of the platform.

5.1 Extracted Personal Data

We determined our data subject by picking a random user from the top 0.01% users sorted by highest reputation. Our data subject has posted more than 90 questions, 9000 answers, and 25,000 comments.

[3] https://github.com/UBA-PSI/ExPD
[4] https://www.selenium.dev/
[5] https://github.com/neo4j/neo4j

The data subject is assumed to be interested in all their answers, questions, and comments on the platform. To allow for an unambiguous comparison of ExPD and API, we will only extract the times of creation, last activitiy, and last edit associated with these entities.

5.2 Extraction with API

One co-author, an experienced software developer, familiarized himself with the Stackoverflow API, obtained an API key, and implemented a Python script that complied to the prescribed rate limit. The whole process took less than two hours. The execution time of the script was about 75 minutes, resulting in 38,739 extracted timestamps.

5.3 Extraction with ExPD

Before recording rules with ExPD, one co-author familiarized himself with the structure of Stackoverflow and the location of the desired timestamp. He decided to take the *profile page* of the data subject as the start-state by putting its URL into ExPD's configuration file. Additionally, he specified the username of the data subject as a keyword in the configuration file. After that, he launched ExPD's learning phase, which opens an instrumented browser. Then, the operator subsequently enters actions in the provided overlay to gather rules for the automatic extraction.

In total, the operator defined eight interaction rules and eight extraction rules. The training required six minutes of manual work, then ExPD started to extract the data automatically (extraction phase), which took 91 minutes. In the subsequent refinement phase the operator worked through the statetypes and extraction rules. After selecting the desired text for each rule, he obtained a CSV file with all exported data. In this case, refinement took three minutes. ExPD extracted 29,401 date values in total.

5.4 Comparison

First of all, we note that ExPD extracted a strict subset of the ground-truth data, i. e., ExPD did not extract data pertaining to other users. However, ExPD only extracted 75.8% of the timestamps contained in the ground-truth. We investigated the missing values and found that we failed to define an extraction rule for the edit-date of a user's answers. None of the pages that the operator encountered before ExPD started the automated extraction contained an edit-date.

To address this issue, which is a central issue in any programming-by-example tool, we plan to give the operator additional control to increase page coverage in the learning phase. Apart from that, ExPD managed to extract all the remaining dates without manual intervention or

programming on the part of the operator. While the API script was faster, the time of manual interaction with ExPD was shorter than the time needed to implement the API script.

6 Discussion

The ExPD prototype is capable of extracting personal data from web applications without programming experience. It takes less human effort than a manual extraction, avoids extracting critical data, and performs all tasks required from crawling to PDF exporting to ease its integration in a process. Because of this, ExPD is a viable option for extracting personal data in a subject access process. For private individuals, ExPD encourages them to do their own research. If an individual is interested in its personal data on a website, ExPD lessens manual labor that this individual might not be willing to do. Overall, ExPD intends to make personal data more accessible for private individuals and support data controllers to comply with the GDPR.

Ethical Considerations With ExPD, the personal data of any data subject in web applications may be extracted. Therefore, ExPD could also be used to gather personal data without legitimate interest or the data subject's consent. However, especially when using an API, extracting personal data is feasible without ExPD, nevertheless. Instead, ExPD at least grants private individuals the possibility to access their public personal data with reasonable effort. Individuals can use this data to understand what others can find out about them.

Limitations Our approach has several limitations.

The web application may not act as defined in the statetype model because of several reasons. Firstly, technical issues like e. g., software errors can cause a state to differ from an expected state. Secondly, the assumption that states of certain statetype share the same interaction options, structure, and data may not always apply. The assumption does not hold if, e. g., a state only includes a button if certain data is present. Thirdly, the application may change during extraction. Especially, if the operator is not in control of the application's deployment, A/B testing may interfere in the extraction. To mitigate these, ExPD checks if elements include similar text before executing the interaction. To make the extraction less vulnerable to these limitations, ExPD could adopt more sophisticated element localisations with the XPath language.

Additionally, ExPD is limited by the operator's knowledge about the web application. ExPD will not extract personal data if the operator does not know where to find personal data in the application. To avoid missing personal data in this way, ExPD could autonomously crawl the application to find applications states containing similar data to already extracted data. ExPD could then point the operator to these application states.

Future Work We intend to make the applications' graph models portable and ultimately distributable. Distributed models can be learned, distributed publicly, and eventually used or improved by a community of users. Consequently, operators can save time using these models and are less prone to make errors during learning.

Additionally, we will reduce missed statetypes in the learning phase by providing suggestions for states that contain potential desired data.

7 Conclusion

ExPD instruments a web browser to extract personal data from web applications. Its architecture is more lightweight than sophisticated Robot Process Automation (RPA) or Record-Replay systems presented in the literature. Thus, ExPD can be set up on desktop machines with little effort.

Operators do not need programming skills to extract data with ExPD. They specify interaction and extraction rules by browsing the web application and labelling the relevant parts of the page via a control panel that floats on top of websites. Rules can be conditioned to ensure that only personal data that is relevant to a particular user is extracted. Using parametrization, a given rule set can be reused for Subject Access Requests (SARs) of different users.

As illustrated in the StackExchange case study, ExPD can also be used by *data subjects*, e. g., when a controller fails to provide a satisfactory response to a SAR. In such cases, ExPD allows data subjects to get an overview of their *public* data footprint on the respective site, which may encourage them to review and amend their profile on that site.

Bibliography

[AC11] Andrica, Silviu; Candea, George: WaRR: A tool for high-fidelity web application record and replay. In: DSN 2011, Proceedings of the 2011 IEEE/IFIP International Conference on Dependable Systems and Networks. IEEE Compute Society, pp. 403–410, 2011.

[AD18] Ausloos, Jef; Dewitte, Pierre: Shattering One-Way Mirrors. Data Subject Access Rights in Practice. Data Subject Access Rights in Practice. International Data Privacy Law, 8(1):4–28, 2018.

[Ch15] Chasins, Sarah; Barman, Shaon; Bodík, Rastislav; Gulwani, Sumit: Browser Record and Replay as a Building Block for End-User Web Automation Tools. In (Gangemi, Aldo; Leonardi, Stefano; Panconesi, Alessandro, eds): WWW 2015, Proceedings of the 24th International Conference on World Wide Web Companion Volume. ACM, pp. 179–182, 2015.

[CMB18] Chasins, Sarah E.; Mueller, Maria; Bodík, Rastislav: Rousillon: Scraping Distributed Hierarchical Web Data. In (Baudisch, Patrick; Schmidt, Albrecht; Wilson, Andy, eds): UIST 2018, Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology. ACM, pp. 963–975, 2018.

[Fi18] Fieldfisher: , A novel approach to data subject requests. https://www.fieldfisher.com/en-ie/locations/ireland/ireland-blog/a-novel-approach-to-data-subject-requests, 2018.

[Fu14] Furche, Tim; Gottlob, Georg; Grasso, Giovanni; Guo, Xiaonan; Orsi, Giorgio; Schallhart, Christian; Wang, Cheng: DIADEM: Thousands of Websites to a Single Database. Proc. VLDB Endow., 7(14):1845–1856, 2014.

[Ga20] Gartner: , Gartner Identifies the Legal and Compliance Technologies to Focus on Post COVID-19. https://www.gartner.com/en/newsroom/press-releases/2020-10-05-gartner-identifies-the-legal-and-compliance-technologies-to-focus-on-post-covid-19, 2020.

[HL16] Herrmann, Dominik; Lindemann, Jens: Obtaining personal data and asking for erasure: do app vendors and website owners honour your privacy rights? In (Meier, Michael; Reinhardt, Delphine; Wendzel, Steffen, eds): Sicherheit 2016: Sicherheit, Schutz und Zuverlässigkeit, Beiträge der 8. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI). volume P-256 of LNI. GI, pp. 149–160, 2016.

[KLH20] Kröger, Jacob Leon; Lindemann, Jens; Herrmann, Dominik: How do app vendors respond to subject access requests? A longitudinal privacy study on iOS and Android Apps. In (Volkamer, Melanie; Wressnegger, Christian, eds): ARES 2020, Proceedings of the 15th International Conference on Availability, Reliability and Security. ACM, pp. 10:1–10:10, 2020.

[Li09] Lin, James; Wong, Jeffrey; Nichols, Jeffrey; Cypher, Allen; Lau, Tessa A.: End-user Programming of Mashups with Vegemite. In (Conati, Cristina; Bauer, Mathias; Oliver, Nuria; Weld, Daniel S., eds): IUI 2009, Proceedings of the 14th International Conference on Intelligent User Interfaces. ACM, pp. 97–106, 2009.

[Lo18] Lockard, Colin; Dong, Xin Luna; Shiralkar, Prashant; Einolghozati, Arash: CERES: Distantly Supervised Relation Extraction from the Semi-Structured Web. Proc. VLDB Endow., 11(10):1084–1096, 2018.

[LRNS02] Laender, Alberto H. F.; Ribeiro-Neto, Berthier A.; Silva, Altigran Soares da: DEByE – Data Extraction By Example. Data Knowl. Eng., 40(2):121–154, 2002.

[MAvE18] Mahieu, René L. P.; Asghari, Hadi; van Eeten, Michel: Collectively exercising the right of access: individual effort, societal effect. Internet Policy Review, 7(3), 2018.

[Ne15] Netravali, Ravi; Sivaraman, Anirudh; Das, Somak; Goyal, Ameesh; Winstein, Keith; Mickens, James; Balakrishnan, Hari: Mahimahi: Accurate Record-and-Replay for HTTP. In (Lu, Shan; Riedel, Erik, eds): USENIX ATC 2015, USENIX Annual Technical Conference. USENIX Association, pp. 417–429, 2015.

[Ur19] Urban, Tobias; Tatang, Dennis; Degeling, Martin; Holz, Thorsten; Pohlmann, Norbert: A Study on Subject Data Access in Online Advertising After the GDPR. In (Pérez-Solà, Cristina; Navarro-Arribas, Guillermo; Biryukov, Alex; García-Alfaro, Joaquín, eds): DPM 2019 and CBT 2019, Proceedings of the ESORICS 2019 International Workshops on Data Privacy Management, Cryptocurrencies and Blockchain Technology. volume 11737 of Lecture Notes in Computer Science. Springer, pp. 61–79, 2019.