



Universiteit
Leiden
The Netherlands

Web privacy measurement in real-time bidding systems. A graph-based approach to RTB system classification

Eijk, R.J. van

Citation

Eijk, R. J. van. (2019, January 29). *Web privacy measurement in real-time bidding systems. A graph-based approach to RTB system classification*. Retrieved from <https://hdl.handle.net/1887/68261>

Version: Not Applicable (or Unknown)

License: [Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden](#)

Downloaded from: <https://hdl.handle.net/1887/68261>

Note: To cite this publication please use the final published version (if applicable).

Cover Page



Universiteit Leiden



The following handle holds various files of this Leiden University dissertation:

<http://hdl.handle.net/1887/68261>

Author: Eijk, R.J. van

Title: Web privacy measurement in real-time bidding systems. A graph-based approach to RTB system classification

Issue Date: 2019-01-29

3

From data collection to graph analysis

This chapter addresses RQ1: how do we move from data collection to graph analysis? It means that we aim to find a path from collection to analysis. Therefore, we start at identifying methodologies for collecting research data. Each methodology is based on three key elements. They are the development of

- (1) software automating a browser in a way that it simulates an end-user visiting a website,
- (2) a proxy capturing and retaining HTTP data flows transmitted between the browser and the website, and
- (3) software managing the crawling movements of a large set of websites in a reliable way, e. g., by a queue manager.

The methodological basis including the three key elements has led to OpenWPM, the generic experimental framework (see Englehardt et al. [2014]).

To understand the real impact of the experimental framework on end-user privacy, we direct our (academic) focus on the transition path from *data collection* to *data analysis*. Therefore, we develop in this chapter a novel Graph-Based Methodological Approach (GBMA) to identify web tracking in HTTP header fields. Our approach consists of three parts.

PART 1. Data collection

PART 2. Data reduction

PART 3. Data modeling

Part 1 and Part 2 correspond with Step 4 in our research methodology (see Section 1.5), viz. to conduct experiments from an end-user perspective and retain the research data. Part 3 corresponds

with Step 5, viz. the development of a metadata model and storing the data into a graph. We discuss the three parts in the sections 3.1 to 3.3. We end this chapter with conclusions (Section 3.4) and we provide an answer to RQ1 (Section 3.5).

3.1 DATA COLLECTION

The first part of the GBMA is data collection. For this purpose, we conduct experiments by simulating an end-user's browser actions and retain the research data. Several key elements are weighed carefully before crawling the web. Below, we will discuss browser data collection (Subsection 3.1.1), data-flow collection (Subsection 3.1.2), and proxy collection (Subsection 3.1.4).

3.1.1 Browser data collection

In this subsection we restrict the scope of our perspective to interactions with services from the desktop. Basically, we take services that can be browsed from a desktop computer as a starting point.

Moreover, we restrict our research to the collection of research data from an end-user viewpoint. In doing so we consider the (desktop) browser as the instrument closest to the end-user.¹⁶¹

There are various ways of collecting research data from an end-user viewpoint. We identify three different ways of collecting research data, i. e., (A) browser centric, (B) server centric, and (C) network centric.

A: BROWSER CENTRIC

A browser centric approach is typically based on either *customizing* the browser itself (henceforth: a browser patch) or *extending* the browser with specialized software (henceforth: a browser extension). Below we provide examples to illustrate the difference between the two approaches.

¹⁶¹ This contrasts with browsing with a smart mobile device. For mobile one can argue that the device itself and the device identifier(s) connected to it are closest to the end-user. The fact that an end-user carries a smart mobile device with him makes it closer to the end-user than a desktop browser.

A1: BROWSER PATCH

Three examples of a specialized browser patches are Prividor [Schütte, 2011], FPDetective [Acar et al., 2013, p. 667], and Privaricator [Nikiforakis, Joosen, & Livshits, 2015]. Local changes were made to the source code, i. e., Prividor and FPDetective modified the Firefox browser to their needs, and Privaricator modified the Chromium browser which is the open-source version of Google's Chrome browser.

A2: BROWSER EXTENSION

Two examples of browser extensions to gather research data are Live HTTP Headers [Savard & Co, 2011],¹⁶² and FourthParty [Mayer & Mitchell, 2012].¹⁶³

Browser patches and browser extensions operate on the level of the browser DOM API and are accessible with JavaScript. The browser DOM API enables access to (1) the HTML DOM which contains cookies [Barth, 2011, RFC 6265] and (2) browser (pre-)loads of resources [Grigorik, 2015]. Below we focus on browser extensions.

FIVE BROWSER OBJECTS

The DOM API gives access to browser objects such as (1) window, (2) navigator, (3) screen, (4) location, and (5) document. We note that the browser object location is not the same as geolocation. It means the originating web server. In Table 3.1 we provide a brief description of the five browser objects. Three of them (window, navigator, and document) have properties.¹⁶⁴ The total number of properties is six (see Table 3.1, next page). Two of them (screen and location) do not have properties in our table.¹⁶⁵ The description illustrates the capabilities to process data through DOM APIs

¹⁶² See, e.g., Krishnamurthy and Wills [2009a] or Van Eijk [2011b] who used the Live HTTP Headers extension to collect their WPM-research data.

¹⁶³ [Mayer & Mitchell, 2012] presented the FourthParty web measurement platform. Others used the browser extension to collect research data, e.g., Eubank et al. [2013]. FourthParty inspired OpenWPM [Englehardt et al., 2014] and the creation of the browser extension 'Tracking Report Card' [Adams, Fedor, Felten, Felten, Fenton, Mayer, Ralli, Toubiana, Turner, & Van Eijk, 2012].

¹⁶⁴ Throughout the thesis we use Uniform Resource Identifier (URI) for a link on a webpage and Universal Resource Locator (URL) for the domain of a website.

¹⁶⁵ Properties of screen are, e.g., height, width. URL: https://www.w3schools.com/jsref/obj_screen.asp (19 February 2018). Properties of location are, e.g., host,

of modern browsers.¹⁶⁶ First of all, we remark that the spelling of ‘referrer’ in the browser object `document.referrer` originates from a misspelling in [Berners-Lee, Fielding, & Frystyk, 1996, RFC 1945, pp. 44–45]. Furthermore, we provide adequate references. The main publication is by Powers [2008], but also others deserve credits, e. g., Van Kesteren, Gregor, Ms2ger, Russel, and Berjon [2015, DOM4], Le Hors, Le Hégaret, Wood, Nicol, Robie, Champion, and Byrne [2004, DOM3 Core], Hickson, Berjon, Faulkner, Leithead, Doyle Navara, O’Connor, and Pfeiffer [2014, HTML5], WHATWG [2015, HTML Living Standard], and Berners-Lee, Fielding, and Masinter [2005, RFC 3986]).

Table 3.1: JavaScript and the HTML DOM.

BROWSER OBJECT	DESCRIPTION
<code>window</code>	Represents the open window in a browser.
<code>window.history</code>	The URI of the HTML pages visited by the end-user.
<code>window.name</code>	The property sets or returns the name of the window.
<code>navigator</code>	Information about the browser.
<code>navigator.userAgent</code>	The end-user agent string of the browser.
<code>screen</code>	Information about the screen of the device.
<code>location</code>	The URI of the current HTML page visited by the end-user.
<code>document</code>	The entire HTML page that has been loaded.
<code>document.URL</code>	The URI of the current HTML page visited by the end-user.
<code>document.cookie</code>	The cookies for the current HTML page.
<code>document.referrer</code>	The URI of the document that loaded the HTML.

Browser extensions are particularly useful for collecting data from (panel)users who opted-in to sharing their (personal) data. An advantage of the extensions is that it is rather easy to adopt

origin. URL: https://www.w3schools.com/jsref/obj_location.asp (19 February 2018).

¹⁶⁶ For instance, Janc and Zalewski [2014] argued that „it is possible to use `window.name` to store persistent identifiers for a given window: if an end-user deletes all client state but does not close a tab that at some point in the past displayed a site determined to track the browser, re-navigation to any participating domain will allow the window-bound token to be retrieved and the new session to be associated with the previously collected data.”

the system to new requirements of the end-user environment. For instance, in August 2015 Mozilla announced that they changed the browser extension API (called WebExtensions) of the Firefox browser. Originally, WebExtensions was designed to be compatible with the API model of the Chrome browser. The change made it easier for developers of add-ons to build extensions that will work across multiple browsers.¹⁶⁷

Thereafter it was easier to collect data and share it with others under predefined conditions due to the enhanced interoperability between different browsers. Many privacy-browser extensions provide this telemetry. TrackerBlock [Brock, 2012] - a Firefox browser extension - was one of the first in the field of WPM to collect aggregated tracking statistics from its end-users. Three more examples are the well-known extensions Ghostery [Pierce, 2010], Electronic Frontier Foundation (EFF)'s Privacy Badger [Eckersley et al., 2014], and Disconnect [Kennish, Chau, & Oppenheim, 2013].

A disadvantage of loading a browser extension is that it increases the entropy of the browser.¹⁶⁸ High browser entropy increases the risk identification by a (unique) browser fingerprint (cf. Eckersley [2010a], see also, e.g., Starov and Nikiforakis [2017a] or Haga et al. [2017, p. 1,665–1,668].).

Although fingerprinting cannot be prevented, a (partial) remedy to mitigate the risk is to use a default browser without browser extensions (see, e.g., Doty [2015]; Tillmann [2013]). The Tor-browser is an example of a browser that goes to great lengths in mitigating the risk [The Tor Project, 2015].

B: SERVER CENTRIC

A server centric approach requires *direct access* to the server's log files. Server data of this kind is usually hard to get access to. For instance, Yahoo released several large server centric datasets to the public in 2016.¹⁶⁹ Two of these are specifically interesting for RTB researchers, i.e., (a) Data Targeting End-User Modeling [Yahoo, 2014] and (b) Search Marketing Advertiser Bidding Data [Ya-

¹⁶⁷ Cf. Needham [2015].

¹⁶⁸ Supra Section 2.4.

¹⁶⁹ URL: <https://webscope.sandbox.yahoo.com/catalog.php?datatype=a> (30 April 2017).

hoo, 2002]. Both sets contain a sample of advertiser's bid and revenue information.¹⁷⁰

We mention three studies on server centric data: (1) Zhang, Yuan, Wang, and Shen [2014], (2) Budak, Goel, Rao, and Zervas [2014], and (3) Aziz and Telang [2015]. Zhang, Yuan, Wang, and Shen [2014] analyzed a dataset that contained the server-side data of three seasons of the iPinYou global RTB algorithm competition. Budak et al. [2014] analyzed web browsing histories of 14 million individuals. Aziz and Telang [2015] had access to and analyzed a dataset containing over 30 million bid requests for 586,909 unique individuals who are potential customers of an advertiser.¹⁷¹

Here we note, that there are usually almost 60 bid requests stored for each individual. The data is suited for studying end-user behavior from an algorithmic perspective. The set of bid requests contained three particular data properties (a) the amount that was bid for in every bid request, (b) whether the bid was successful, and (c) whether the potential customers made any purchase in the next three days.

C: NETWORK CENTRIC

Unlike the two approaches above, a network centric approach *captures HTTP data flows* transmitted between the browser and the website. The data is retained by specialized software such as Mitmproxy [Cortesi et al., 2010]. Mitmproxy is a proxy written in Python. The software is capable to intercept data transmitted through HTTP Secure (HTTPS).

For our approach Mitmproxy is the basis. This is also true for, e.g., OpenWPM [Englehardt et al., 2014] and MobileScope [Soltani et al., 2012]. Its data format is widely used and contributes to the sharing of data between researchers.

¹⁷⁰ Yahoo [2014] contains „a small sample of advertiser's bid and revenue information over a period of 4 months. Bid and revenue information is aggregated with a granularity of a day over advertiser account id, key phrase, and rank. Apart from bid and revenue, impressions and clicks information is also included." The data is a snapshot of end-user activity data collected at Yahoo servers during the period from 8 April 2014 to 20 July 2014. (cf. Wiley [2017]).

Yahoo [2002] contains „the bids over time of all advertisers participating in Yahoo Search Marketing auctions for the top 1000 search queries collected during the period from 15 June 2002 to 14 June 14 2003."

¹⁷¹ The origin of the iPinYou dataset is the iPinYou DSP.

OTHER FORMATS, E. G., HAR

There exists software parsers to convert the research data into other data formats, e. g., the widely used HTTP Archive (HAR) format [Odvarko, Jain, & Davies, 2012]. It is important to know that Nassri [2015] provides a list of resources that support HAR. At the time of writing, the list contained 27 applications.

HAR enables further analysis with, e. g., (1) browser-debugging tools for web developers, (2) specialized network-analysis software, or (3) network performance measurement tools such as BigCloud (see, e. g., Nassri [2015] or Grigorik [2013]; Kahle [1996]). BigCloud is a cloud service operated by Google. The service enables researchers to upload a HAR dataset into a Dremel database [Melnik, Gubarev, Long, Romer, Shivakumar, Tolton, & Vassilakis, 2010] and access their data via a Representational State Transfer (RESTful) API with SQL-like queries. The R-package Bigrquery [Citro, 2014] bridges BigCloud with R's statistical toolbox [R Core Team, 2014]).

The Mitmproxy data can easily be parsed into smaller subsets. This (1) enables prototyping and (2) makes everyday data juggling with big crawls considerably easier. For instance, a small set of research data - e. g., 10% of all the data records - is usually sufficient for prototyping and testing different data models.

At the end of this paragraph on network centric, we would like to communicate one lesson learned from a change in the Mitmproxy data format.¹⁷² The Mitmproxy change led to a problem in parsing a crawl from January 2013. The crawl was collected with Mitmproxy version 0.8 and a recent crawl 2 with version 11.2. As it turned out the results of the two crawls were no longer comparable. We solved the issue by parsing the historic dataset with a corresponding version of Mitmproxy.¹⁷³

3.1.2 *Data-flow collection: existing methods*

For data-flow collection (based on the network centric approach) we need specialized software. As a case in point, we mention the software testing framework Selenium [Selenium Project, 2004] that enables the collection of data flow by automating the Fire-

¹⁷² Cf. the change in the browser extension API.

¹⁷³ To revert to a specific version of Mitmproxy on Ubuntu we used the following command: `sudo apt-get install package=<version>`.

fox browser. In this environment Van Eijk and Terra [2012] automated website visits with a Python script. Before we discuss our approach to data-flow collection for our first experiments, we remark that other ways of gathering data flows exist. Below we give four examples based on specialized web development tools designed for web developers to test and debug their code: (A) Python scripting library, (B) PhantomJS, (C) debugging protocol of the Chrome browser, and (D) W3C Webdriver. Then we continue with (E) our 3-step procedure of data-flow collection for our *first experiments* of which the advantages are also described. We note that our procedure of data-flow collection for our *later experiment* for the purpose of our empirical view of RTB systems (see Section 4.4) is described in Subsection 4.4.2.

A: PYTHON SCRIPTING LIBRARY

Researchers and browser-extension developers can use a scripting library such as Python's Requests module (see, e.g., Lavrenovs and Melon [2018] or Reitz, Benfield, Cordasco, and Petrov [2011]).¹⁷⁴ Scripting with a module capable of making HTTP requests is a quick way of automating web visits to collect research data.¹⁷⁵ One important thing to keep in mind when experimenting with different scripting libraries is to take care of support for cookies. That is, whether the session and the persistent cookies are retained. Support for cookies is important for end-user impersonation, which we will discuss in more detail in Subsection 3.1.4.

B: PHANTOMJS

Many researchers use PhantomJS [Hidayat, 2011], a headless browser. The scriptable JavaScript API makes it fit for the purpose of data-flow collection and for WPM (see, e.g., Some, Rezk, and Bielova [2018] or Hannak, Sapiezynski, Molavi Kakhki, Krishnamurthy, Lazer, Mislove, and Wilson [2013]). PhantomJS is built on WebKit which is also the rendering engine for, e.g., the Safari browser [Apple, 2005]. Recently, the browser vendors Mozilla [Claypotch, 2017] and Google [Bidelman, 2017] announced that

¹⁷⁴ The source code of the module is available on GitHub. URL: <https://github.com/kennethreitz/requests> (15 December 2015).

¹⁷⁵ An example of a simple Python command to visit a web page is, e.g., `resp = requests.get('http://campusdenhaag.nl/crk')`.

their browsers are ready for headless usage. The announcement is good news for WPM scholars, since it enables them to use headless Firefox or Chrome as a substitute for PhantomJS. We remark that Linux already offered a workaround for data collection in ‘headless mode’ by using virtual framebuffers.

C: DEBUGGING PROTOCOL

Many browser-extension developers use the (remote) debugging protocol for the Chrome browser, which is part of the Chrome Developer Tools [Google, 2015]. The protocol is scriptable with Python. Clift [2014] deploys the protocol by sending JSON messages [Bray, 2014, RFC 7159] via a network connection to enable the automation of a browser instance. This approach is particularly interesting for WPM researchers who are looking to conduct experiments on a smart phone, tablet, or another mobile device. Finally, we remark that the early version of Netograph [Cortesi, 2011] was built on Mitmproxy [Cortesi et al., 2010]. Recently, the codebase of Netograph [Cortesi, 2017] has been brushed off and changed to the Webdriver debugging protocol [Stewart & Burns, 2018]. It has a main focus on (1) large scale continuous crawling, (2) storing processed metadata in Google Bigtable [Google, n.d.-a], and (3) providing API access for further analysis.¹⁷⁶

D: W3C WEBDRIVER

The Webdriver protocol enables a WPM researcher to automate site visits.¹⁷⁷ Similar to (1) Selenium [Selenium Project, 2004] and (2) the Chrome debugging protocol, W3C Webdriver [Stewart & Burns, 2018] is an attempt to standardize the control and introspection of browsers.

3.1.3 *Our 3-step procedure of data-flow collection*

Now that we discussed four different ways of automating the browser, we will deepen the understanding of our GBMA to collecting flow data (i.e., Part 1 of the GBMA). For our first exper-

¹⁷⁶ URL: <https://netograph.io/docs/api/docs> (27 August 2018).

¹⁷⁷ Stewart and Burns [2018]: „WebDriver is a remote control interface that enables introspection and control of user agents. It provides a platform- and language-neutral wire protocol as a way for out-of-process programs to remotely instruct the behavior of web browsers.”

iments we used the Alexa website. We are aware that many researchers use other datasets, e.g., the ranking of Quantcast¹⁷⁸ or SEOmoz.¹⁷⁹ Our choice was guided by the quality of the results of the crawls (see Table 3.2). We start discussing three collection steps (called C-steps). The three C-steps are performed in accordance with the experimental framework that is fit for long-tail experiments.

C-STEP 1: FILTERING ALL .NL WEBSITES

First, we defined our crawlset by filtering all .NL websites from the top1m dataset of that day.¹⁸⁰ We provide Listing 3.1 as an example of such a filter.

Listing 3.1: Simple Python script to select a crawlset from Alexa top1m [Van Eijk & Terra, 2012].

```

1 from urllib2 import urlopen
2 from lxml import html
3
4 for n in range(20):
5     d = html.fromstring(urlopen("http://www.alexa.com/
        topsites/countries;%d/NL" % (n, )).read())
6     for site in d.cssselect("div.desc-container h2 a"):
7         print site.get("href").split("/")[-1]
```

The script starts with a declaration of two helper modules (Listing 3.1, rr. 1–2). Because the top ranking .NL websites are not listed on one long page, but on 20 pages, we need to loop through the Alexa website to collect our selection (Listing 3.1, rr. 4–5). We used the HTML tag ‘div.desc-container’ as an easy way to parse the HTML page (Listing 3.1, r. 6). The last line makes sure the output is written in a pretty format (Listing 3.1, r. 7).

178 Quantcast provides another popular dataset that ranks websites based on the number of people in the Netherlands, who visit a website within a period of time. URL: <https://www.quantcast.com/top-sites/NL> (20 July 2015).

179 SEOmoz provides the Moz Top 500, a list of domains ranked by the number of linking root domains. URL: <https://moz.com/top500> (31 march 2018).

180 Top1m means the first one million websites of that day. URL: <http://www.alexa.com/topsites/countries/NL> (14 January 2016).

C-STEP 2: CONSTRUCTING A REPRESENTATIVE CRAWL

Second, we started to construct the crawl with

- (1) a fresh browser instance,
- (2) empty cache, and
- (3) an empty profile, i. e., neither browsing history nor cookies (henceforth: first view).

Each domain on the list of target websites was visited twice but not in the same order. We randomized the list before visiting the sites for the first time (henceforth: round 1), and randomized the list again before visiting the sites for the second time (henceforth: round 2). No links were clicked or followed. No ads were clicked. Moreover, we also did not log in to any (social media) services or click on any social media buttons. In later experiments we performed more crawls (up to six crawls).

We remark that websites usually do not publish a social widget on their homepage which makes detecting the trackers less effective with our type of crawl. To measure *tracking by social widgets*, a deep crawl method is needed. The issue of end-user impersonation will be discussed in Subsection 3.1.4.

C-STEP 3: CRAWLING A LARGE SET IN A RELIABLE WAY

Third, Van Eijk and Terra [2012] used a queue manager to manage crawling a large set of websites in a reliable way. The script is written in Python. A queue manager is necessary. The queue manager has three functions,

- (1) to load the set of websites, i. e., performing round 1 and then round 2, one by one,
- (2) to overcome browser crashes and (programming) problems restraining the browser to finish loading the entire document, and
- (3) to maintain the browser state by retaining the Firefox profile. The profile contains important end-user data, i. e., browsing history and cookies.

Without a queue manager the end-user data could be lost in the event of a browser crash or when a page-load problem occurs. To prevent this loss from happening, the queue manager retains

the Firefox profile and subsequently terminates the Firefox browser instance. Before launching a new Firefox instance, it restores the Firefox profile, such that the crawl can continue where it was interrupted.

We continue our research in five paragraphs, viz. (A) the first experimental results; (B) the results analyzed; (C) the long tail experimental results; (D) the impact of technical measures and (E) conclusions on data-flow collection.

A: THE FIRST EXPERIMENTAL RESULTS

For the crawl itself, we used a shallow crawl method instead of a deep crawl.¹⁸¹ Below we provide a definition for shallow crawl (Definition 3.1) and deep crawl (Definition 3.2).

DEFINITION 3.1: A SHALLOW CRAWL is a process of visiting the top domain of each website in a list of websites.

DEFINITION 3.2: A DEEP CRAWL is a process of visiting the top domain and/or visiting (a number of) internal links of each website in a list of websites.

We compiled our list of websites by selecting all the Dutch websites (.NL top level domain) from the Alexa [n.d.] top 1,000,000 list of websites in the world (henceforth: top1m dataset). For the shallow crawls (listed in Table 3.2) each domain name in the crawl list was visited without visiting (deep) links.

Table 3.2 shows the number of .NL websites in the top1m dataset for six different crawls. There we provide a numerical summary of the six crawls with the measures of meaning in the third column 'Numerical' and the corresponding result in the fourth column 'Summary'.

From the numerical data in Table 3.2 we may conclude that the average number of the most popular websites in the Netherlands is about 8,000 (to be precise, the mean value is 8,277) as considered with respect to the top1m popular websites in the world. To put these numbers into perspective, the number of registered .NL

¹⁸¹ See also, e.g., Hoofnagle and Good [2012a; 2015] for the difference between shallow and deep crawling.

websites in 2014 was over 5 million (to be precise: 5,531,186) according to the annual report of SIDN [2014], the Dutch top-level domain administrator.

Table 3.2: Numeric summary and total number of six different crawls of .NL websites in the Alexa top 1,000,000 list of popular websites.

CRAWL	.NL WEBSITES	NUMERICAL	SUMMARY
Crawl1 (Nov 2012)	8,472	Min.:	7,968
Crawl2 (Dec 2012)	8,067	1st Qu.:	8,094
Crawl3 (Jan 2013)	7,968	Median:	8,294
Crawl4 (Apr 2013)	8,415	Mean:	8,277
Crawl5 (Mar 2015)	8,174	3rd Qu.:	8,458
Crawl6 (Jun 2015)	8,567	Max.:	8,567

B: THE RESULTS ANALYZED

Despite a clear selection criterion to compile our crawl set, the total number of Dutch websites in Table 3.2 differs from other sources. The difference can be explained by looking closer at the Alexa’s ranking methodology (See Alexa [2015a]. Below we will briefly discuss (B1) Alexa’s ranking methodology and (B2) the importance of the long-tail phenomenon for WPM.

B1: ALEXA’S RANKING METHODOLOGY

Alexa collects data from panel members who opted in to the sharing of their browsing data. Alexa counts who makes actual visits to which websites in a rolling three month period. The ranking of a website is based on two elements, (1) the number of panelists who visit a website on a given day and (2) the number of page views by the panelists.

However, websites are ranked relatively to other websites. Therefore, changes in traffic to one website affect the ranking of others. This is also known as a long-tail phenomenon (cf. Alexa [2015b]), which we will discuss below (see B2 and C).

B2: THE IMPORTANCE OF THE LONG-TAIL PHENOMENON FOR WPM

Long-tail phenomena are well studied in other (academic) fields.¹⁸² However, this is not the case for WPM, although his conclusions of WPM studies are often based on data collected from rankings of the top-websites.¹⁸³ Therefore, it is - from a WPM-researcher's viewpoint - important to understand the long-tail phenomenon of ranking. It helps us to understand (1) what is happening when we crawl a set of websites and (2) whether different crawls can be used in longitudinal studies.

C: THE LONG TAIL EXPERIMENTAL RESULTS

In the generic experimental framework, the choice for using (parts of) datasets such as Quantcast's ranking and Alexa's top1m for crawling has been guided by a personal preference.¹⁸⁴ Table 3.2 shows six examples of crawls of .NL websites. The long-tail phenomenon for .NL websites is illustrated in Figure 3.1.

The histogram of Figure 3.1 presents (a) the number of .NL websites (topnl) in relation to the Alexa rank, (b) the percentage of .NL websites (percentnl) relative to their ranking, and (c) the percentage of .COM websites (percentcom) relative to their ranking. The top1m dataset of 22 June 2015 contained 486 .NL websites ranked under ranking 100,000. This number counts for about 5% of the .NL websites in the top1m dataset. The remaining 95% of .NL websites ranked 100,000 or lower and are subject to the long-tail phenomenon. More than half of the .NL websites rank very low, i. e., ranked 524,289 or lower.

Furthermore, Figure 3.1 shows that .NL websites are slightly (about 6%) more prone to the long-tail phenomenon than .COM websites. Websites with a low ranking, are much more sensitive

¹⁸² See also, e.g., Anderson [2007] for a detailed description of the long-tail phenomenon. Anderson [2007] examined the three questions: (1) „What happens when there is almost unlimited choice?“, (2) „What happens when everything becomes available to everyone?“, and (3) „When does the combined value of the millions of items that only sell in small quantities equal or even exceed the value of a handful of best-sellers?“.

¹⁸³ See also, e.g., Papadopoulos, Kourtellis, and Markatos [2018] or Scheitle, Jelten, Hohlfeld, Ciprian, and Carle [2018]: „The conclusions of *security* studies are often based on data collected from domains that appear on commercial rankings of the 'top' websites. However, the data sources and methodologies used to compile these rankings vary widely and their details are unknown, leading to hidden properties and biases in the rankings.“ (Scheitle et al. [2018, p. 13], emphasis added).

¹⁸⁴ Supra n. 178.

to small increases or decreases in traffic. The long-tail ranking applies to both .NL and .COM websites. Straightforwardly stated, websites ranked 100,000 or lower can be identified ranking in the long-tail of top1m websites.

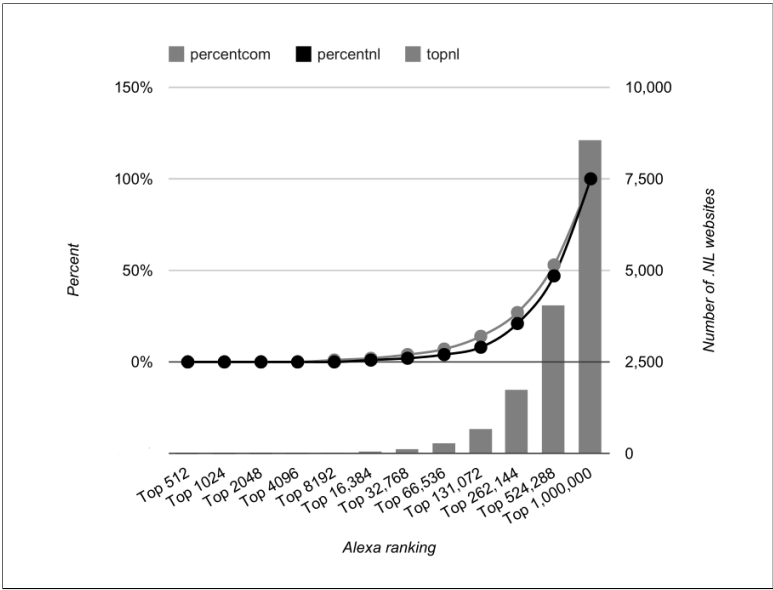


Figure 3.1: Histogram of (a) the number of .NL websites (topnl) in relation to the Alexa rank, (b) the percentage of .NL websites (percentnl), and (c) the percentage of .COM websites (percentcom). The percentages are relative to the total websites, i.e., respectively 8,567 .NL websites and 554,768 .COM websites. The Alexa global top1m dataset was retrieved 22 June 2015, from URL: <http://s3.amazonaws.com/alexa-static/top-1m.csv.zip>.

D: THE IMPACT OF TECHNICAL MEASURES

We note that data-flow collection will become harder as technical measures to mitigate the risk of State surveillance improve (see, e.g., Kranch and Bonneau [2015]). The aim of the risk measures is to mitigate a breach of confidentiality of the communication. Obviously, this has an impact on WPM.

There are two clear drawbacks to the measures to mitigate the risk: (1) the measures make it harder to collect research data through web crawls, and (2) the measures restrict the transparency and ability to verify the marketing technology. Three examples of the application of relevant technical measures to WPM are:

- (1) HTTP Strict Transport Security (HSTS) [Hodges, Jackson, & Barth, 2012, RFC 6797],
- (2) public key pinning extension for HTTP [Evans, Palmer, & Sleevi, 2015, RFC 7469], and
- (3) Transport Layer Security (TLS) Error Reports [Moeller & Langley, 2015, RFC 7507].

Below we give a brief characterization of these three measures.

- Ad 1. HSTS is a measure enabling a web site and a user agent to interact only over secure connections. It prevents piggyback on commercial tracking cookies to identify end-users.
- Ad 2. Public-key pinning is a measure to counter man-in-the-middle attacks due to compromised Certification Authorities.¹⁸⁵ Technically, Mitmproxy cannot intercept pinned data flows once the original certificates are retained in a browser.
- Ad 3. Firefox checks server configurations for insecure connections due to protocol downgrade attacks on depreciated versions of TLS.¹⁸⁶ This behavior may result in websites not loading during a crawl.

E: CONCLUSIONS ON DATA-FLOW COLLECTION

Data-flow collection will become harder as technical measures to mitigate the risk of state surveillance will be stricter. These developments have an impact on data-flow collection according to our GBMA approach. So, collecting data with the Chrome debugging protocol has the potential to become the default approach for collecting data for WPM researchers. In all, we have three conclusions.

¹⁸⁵ [Hodges et al., 2012, RFC 6797] defines a new HTTP header that allows „web host operators to instruct user agents to remember (‘pin’) the hosts’ cryptographic identities over a period of time. During that time, user agents will require that the host presents a certificate chain including at least one Subject Public Key Info structure whose fingerprint matches one of the pinned fingerprints for that host.”

¹⁸⁶ Firefox message: Error code: ssl_error_unsupported_version URL: <https://support.mozilla.org/en-US/kb/tls-error-reports> (10 July 2015).

CONCLUSION 1: In conjunction with Table 3.2 we may conclude that filtering .NL websites from the top1m dataset on any given day with the help of the GBMA results in a crawl set fit for studying web tracking despite the long-tail effect.

CONCLUSION 2: Because .COM websites account for the overall majority of websites ranked in the daily top1m dataset, we may conclude that the long-tail phenomenon to take into account is a maximum of 6% of the researcher's crawl set.

CONCLUSION 3: If the study is longitudinal, then datasets over time will not be linkable in its entirety assuming that the domain name is the key to link the datasets.

3.1.4 *Proxy collection*

The goal of proxy collection is end-user impersonation. Our approach to automate the browser was conducted from an end-user viewpoint with the latest version of Firefox at the time of crawling. According to our methodology for the collection of research data we used three key elements (See the introduction of this chapter, viz. (1) automating a browser (2) using a proxy and (3) using a queue manager). The browser was installed to run headless on a minimal installation of the latest version of the Ubuntu server. The data was collected in the Netherlands over a standard consumer Asymmetric Digital Subscriber Line (ADSL)-connection to the internet.

For technical reasons, we made two minimal changes to its default configuration. First, we installed a self-signed certificate to enable the interception of HTTPs because Mitmproxy cannot intercept the protocol without it.¹⁸⁷ Second, for the same reason, we disabled support for SPDY, which is a network protocol that uses header compression. HTTP/2 (Peon and Ruellan [2015, RFC 7541], Belshe, Peon, and Thomson [2015, RFC 7540]) is replacing SPDY. Yet, the configuration of header compression in Firefox is located in the same browser-configuration subkeys.¹⁸⁸ This subsection consists of two parts, viz. (A) end-user impersonation and (B) data-management issues on the data collection.

¹⁸⁷ Cf. Cortesi et al. [2010].

¹⁸⁸ In Firefox, i.e., 'about:config', the keys to configure HTTP header compression are located under 'network.http.spdy.enabled.http2'.

A: END-USER IMPERSONATION

We took in our experiments the following seven elements into account: (A1) end-user events, (A2) end-user profiling, (A3) localization, (A4) fingerprinting, (A5) bot detection, (A6) end-user consent, and (A7) forensic screenshots. These elements are closely related to the online behavioral patterns of the end-user. In effect, we see that behavior is often repeated. This leads to diversification into distinct patterns, e. g., visiting favorite websites in a specific order around the same time each day. For now, we are only interested in the phenomenon of proxy collection. Therefore, we disregard a classification that is too detailed. Below we give a straightforward characterization of the seven elements A1 to A7.

A1: END-USER EVENTS

Each end-user event is straightforward. No JavaScript emulation of mouse movement and/or scrolling is used to refine the impersonation of an end-user. Many complex end-user interactions can be emulated by the Selenium Project [2004] end-user facing API.¹⁸⁹ An example of end-user interaction required by default by the Firefox browser is ‘click to play’. The feature is a recent security measure to reduce malware infections based on (known) vulnerabilities of the browser add-ons, e. g., the Adobe Flash player. Technically, the browser inserts a wrapper that requires end-user interaction before the content is passed to a browser add-on.¹⁹⁰

A2: END-USER PROFILING

End-user profiling is straightforward. No (pre-)training of a (browser) profile is used to impersonate an end-user by visiting specific categories of websites.¹⁹¹ We crawled with an empty browser state. In contrast, Van Eijk [2011b] visited Dutch children’s websites three days prior to collecting research data, which resulted

¹⁸⁹ I.e., the JavaScript Object ‘org.openqa.selenium.interactions.Actions’. Selenium gives a researcher many options to simulate end-user interactions, e.g., a mouse wiggle. See URL: <http://selenium.googlecode.com/git/docs/api/java/org/openqa/selenium/interactions/Actions.html> (15 January 2016).

¹⁹⁰ Firefox versions 8 and above support click-to-play. In Firefox, i.e., in Firefox’s configuration settings (‘about:config’) the key to configure click-to-play is located under ‘plugins.click_to_play’.

¹⁹¹ Supra Definition 1.9.

in an advertisement for a Donald Duck subscription in Dutch on a German news website.

A big challenge with this approach is to keep the profiles clean. The risk of polluting the profile accidentally is real which means that it is difficult to maintain a *chain of evidence* (a term used in computer forensics related to the methodology of managing evidence that is destined for use in court).

A3: LOCALIZATION

Although a localized Dutch version of the Firefox browser exists, i.e., 'firefox-locale-nl', we installed the default Firefox browser. In addition, no changes were made to the user agent string or the referrer header.¹⁹² The referrer header is an important field in the HTTP request header [Fielding & Reschke, 2014b, RFC 7231, Section 5] providing information about the request context.¹⁹³

A4: FINGERPRINTING

Except from refraining from browser extensions, e.g., the Flash plug-in that allows end-users to play streaming media, no measures to mitigate browser fingerprinting were taken. An example of a fingerprinting mitigation measure is the automated generation of random browser settings while crawling. We refer to Gulyas, Some, Bielova, and Castelluccia [2018], a recent study on the uniqueness of browsers resulting from the presence of browser extensions.¹⁹⁴

The technology landscape for streaming media is changing. The Flash technology is being replaced by HTML5 due to its greater interoperability capabilities. From a technical perspective, a Flash cookie is not that different from a HTTP cookie. Even though Flash cookies are stored in a different place on the end-user's computer, Flash technology transmits and receives cookies over HTTP. This means that Flash cookies can be collected by a network centric approach as well as with a browser centric approach.

¹⁹² The referrer header can be disabled by changing the Firefox configuration setting 'Network.http.sendRefererHeader'. The default value is '3'. Changing it to a value of '0' causes the Firefox browser to omit a referrer field in the HTTP request header [Fielding & Reschke, 2014b, RFC 7231, Section 5].

¹⁹³ Supra Section 2.5.

¹⁹⁴ Gulyas et al. [2018] analyzed how unique users are based on their behavior. They reported that almost 55% (54.86% to be precise) of the 16,393 user in their dataset that have installed at least one detectable extension are uniquely identifiable.

In fact, Google announced in February 2016 that it is phasing out Flash ads.¹⁹⁵ Google's strategy is allow only display ads in HTML5. Moreover, Mozilla removed support for flash.¹⁹⁶ Oracle's strategy for sites running Java technology was to encourage its implementers to migrate to the Java Network Launching Protocol (JNLP).¹⁹⁷ It is quite in time now that Adobe announced on 25 July 2017 that it will end support for Flash by the end of 2020.¹⁹⁸ Other companies, including Microsoft,¹⁹⁹ Apple,²⁰⁰ and Facebook,²⁰¹ have made similar announcements in response to the changing streaming media landscape.

A5: BOT DETECTION

We must assume that various threat mitigation measures have been put into place to deal with ad fraud. Yet, we did not implement a bot-detection mitigation strategy.

Threat mitigation measures by RTB actors are necessary since publishers and website visitors have been subject to injected malware in ad slots on a web page.²⁰² For instance, Majumdar, Kulkarini, and Ravishankar [2007] investigated click fraud in Content Delivery Networks (CDNs). A CDN is a network provider, e. g., Akamai, a company that facilitates the worldwide distribution of ads. Majumdar et al. [2007] proposed four protocols to mitigate replay and fabrication attacks for CDN providers. We provide two exam-

195 URL: <http://doubleclickadvertisers.blogspot.nl/2016/02/google-display-ads-go-100-html5.html> (25 July 2016).

196 Mozilla announced in 2015 to remove support for most plugins for (streaming) media content, e. g., Microsoft's Silverlight and Oracle's Java by the end of 2016. URL: <https://blog.mozilla.org/futurereleases/2015/10/08/npapi-plugins-in-firefox/> (10 October 2015).

197 URL: http://docs.oracle.com/javase/8/docs/technotes/guides/deploy/applet_dev_guide.html#JSDPG1032 (10 October 2015).

198 URL: <https://blogs.adobe.com/conversations/2017/07/adobe-flash-update.html> (26 July 2017).

199 URL: <https://blogs.windows.com/msedgedev/2017/07/25/flash-on-windows-timeline/#D6TjufFSsIgJ03TZ.97> (26 July 2017).

200 URL: <https://webkit.org/blog/7839/adobe-announces-flash-distribution-and-updates-to-end/> (26 July 2017).

201 URL: <https://developers.facebook.com/blog/post/2017/07/25/Games-Migration-to-Open-Web-Standards/> (26 July 2017).

202 An example of a popular Dutch website that was affected by 'malvertising' is nu.nl. URL: https://www.security.nl/posting/395932/Nu_nl+laat+advertenties+extra+op+malware+controleren (12 December 2015).

ples of specialized software capable of detecting link spam caused by bots crawling websites.²⁰³

First, we mention Bad Behavior [Jaquith, Firas, Error, & Skeltoac, 2014] which analyzes the HTTP headers, IP address, and other metadata regarding HTTP requests to determine bot activity. For instance, if the software detects non-compliance with the HTTP State Management Mechanism [Kristol & Montulli, 2000, RFC 2965], it may ignore the bot and block its traffic. Table 3.3 contains a (partial) Bad behavior bot detection report illustrating common bot detection strategies. The data was captured from a website with a PHP implementation of Bad Behavior.²⁰⁴

Table 3.3: Bad Behavior bot-detection report.

REASON	COUNT	PERCENT
Required HTTP header 'Accept' missing	40	75.47%
Web browser attempted to send a traceback	11	20.75%
False claim. User agent claimed to be Googlebot	1	1.89%
User agent claimed to be MSIE with invalid OS version	1	1.89%
Banned proxy server in use	0	0%
HTTP header 'Connection' contains invalid values	0	0%
HTTP header 'Referer' present but blank	0	0%
IP address found on (external) blacklist	0	0%
Required HTTP header 'Accept-Encoding' missing	0	0%
Rotating user agents detected	0	0%
Prohibited HTTP header 'Range' present	0	0%
Prohibited HTTP header 'via' present	0	0%
Prohibited HTTP header 'Proxy-Connection' present	0	0%
User agent was found on blacklist	0	0%
User agent is required but none was provided	0	0%
Use of rotating proxy servers detected	0	0%

²⁰³ See also the Interactive Advertising Bureau (IABureau)/ABC Spiders & Robots Lists [Audit Bureau of Circulations, 2013]. Other examples are, e.g., HostKarma, Spamhaus, SORBS, Google Safe Browsing, and Yandex Safe Browsing.

²⁰⁴ The bot-detection report was retrieved on 6 July 2015 from our own dokuwiki webserver URL: <http://www.blaeu.com>. To save costs, the domain has been reconfigured to serve email only since 20 July 2015. The bot-detection report has been retained in our lab log on 6 July 2015.

Second, we mention the Project Honey Pot [Unspam Technologies, n.d.] which is, in contrast to Bad Behavior, a distributed spam detection solution. Project Honey Pot provides a completely automated public Turing test to tell computers and humans apart (CAPTCHA) [Rosenberg & Jennings, 2008, RFC 5039]. Recently, CloudFlare - a cloud provider - started using Project Honey Pot to protect their clients from spambots.²⁰⁵

A6: END-USER CONSENT

For the end-user consent the following holds. Neither explicit consent for tracking cookies was expressed, e.g., by agreeing to a cookie statement, nor implied consent was given, e.g., by clicking three or more deep links.

A7: FORENSIC SCREENSHOTS

We did not capture a screenshot of the webpage after rendering, although the Selenium WebDriver includes this functionality. However, for certain types of experiment, e.g., checking compliance with the Dutch Cookie law, the information is relevant. The forensic screenshot serves as evidence of the information provided to the end-user at the time of crawling the webpage.

B: DATA-MANAGEMENT ISSUES ON THE DATA COLLECTION

Above, the necessary description of the elements used in our experiments is given. Other data management issues are stored in accordance with the rules given by the Leiden University and in particular by the Faculty of Law. This ends the first step in our Graph-Based Methodological Approach (GBMA).

3.2 DATA REDUCTION

The second part in our GBMA is data reduction. There, the key objective of the methodology is to retain precisely the research data that we need for our data model. To better understand this

²⁰⁵ See CloudFlare's FAQ. URL: <https://support.cloudflare.com/hc/en-us/articles/203366080-Why-do-I-see-a-captcha-or-challenge-page-Attention-Required-trying-to-visit-a-site-protected-by-CloudFlare-as-a-site-visitor-> (10 July 2015).

part, we embark upon the difference between the two terms data and information. Data are (raw) observations, i.e., in our case the Mitmproxy data. Information is pre-processed data to which meaning is given.²⁰⁶ Reducing the research data requires three reduction substeps (called R-steps).

R-STEP 1: Parse the Mitmproxy data.

R-STEP 2: Ensure the data quality.

R-STEP 3: Retain the small data in a graph.

Below we discuss the three R-steps. We guide the discussion by a basic example of the extraction of HTTP response header fields (see Listing 3.2).

Listing 3.2: Command-line example parsing a Mitmproxy data file (Bash).

```

1  $ cat parse.py
2  def response(context, flow):
3  ext = tldextract.extract(flow.request.host)
4      request_headers = [{"name": k, "value": v} for k, v in flow
5                          .request.headers]
6      response_headers = [{"name": k, "value": v} for k, v in
7                          flow.response.headers]
8      print response_headers
9
10 $ mitmdump -n -q -s parse.py -r <file>.mitm | \
11 $ cut -d : -f 1 | sort | uniq > parsed_headers.csv

```

R-STEP 1: In the first R-step, we parsed the Mitmproxy data with an inline script. This script is based on the 'flow' object included in the 'libmproxy' Python library.²⁰⁷ We used a Python script (Listing 3.2, r. 2–6) to extract the HTTP response headers and piped commands (Listing 3.2, r. 8–9), i.e., the commands are separated by the character '|', to retain the research data. Below we briefly explain the four commands.

- (1) Parse the crawl to extract all HTTP header fields.
- (2) Filter (see 'cut -d : -f 1' which parses out column 1 from a colon (:) delimited input).

²⁰⁶ Cf. Meesters [2014, pp. 83–85], supra Section 1.3.

²⁰⁷ We refer to Cortesi, Shaver, and Wingert [2014] for more examples of inline scripting.

- (3) Sort.
- (4) De-duplication (see 'uniq') of all HTTP header fields.

The outcome is a file (`parsed_headers.csv`) containing a list of distinct HTTP response header fields.

The parser interfaces with the research data through Mitmproxy's API. The flow object contains methods to parse information (a) from the HTTP request header and (b) from the HTTP response header. With Python we extracted typical crawl variables, e.g., referrer, location, user agent, and cookie.²⁰⁸ We remark that the cookies are shaped as tuples by Mitmproxy. A tuple is a sequence of immutable Python objects.²⁰⁹

R-STEP 2: In the second R-step (after the required data is extracted), the data is ready for inspection for incorrect or incomplete values. Browser crashes or page-load problems that occurred during the crawling process may have impacted the quality of the small data.

R-STEP 3: In the third R-step we imported the small data into a data model that is appropriate for graph-based analysis. In our research we use Neo4j, an open-source graph database with access to the research data via a RESTful API [Robinson et al., 2015]. Neo4j comes with a number of language drivers, e.g., Go, Java(Script), .NET, Perl, PHP, Python, R, and Ruby. We experimented with Perl, Python and R bindings. The RNeo4j R package [N. White, 2014] connects Neo4j with R's statistical toolbox which includes the iGraph R package.²¹⁰

CONCLUSION ON DATA REDUCTION

In conclusion, data reduction is a necessary part of GBMA. The aim is selecting small data from collected raw data required for data modeling.

²⁰⁸ See also the respective RFC, i.e., 'Referer' [Fielding & Reschke, 2014b, RFC 7231, Section 5.5.2], Location [Fielding & Reschke, 2014b, RFC 7231, Section 7.1.2], User agent [Fielding & Reschke, 2014b, RFC 7231, Section 5.5.3], and Cookie [Barth, 2011, RFC 6265].

²⁰⁹ URL: http://www.tutorialspoint.com/Python/Python_tuples.htm (15 August 2015).

²¹⁰ See also Csárdi and Nepusz [2014]; R Core Team [2014]

3.3 DATA MODELING

The third part in our GBMA is the development of a *small-data* model. The key objective of the small-data model is to unlock the information defining web tracking within the *big* dataset itself. Below, we will discuss the following notions: a generic web-tracking model (Subsection 3.3.1), the formalization of graph-mining rules (Subsection 3.3.2), an extended web-tracking model (Subsection 3.3.3), high-entropy identifiers (Subsection 3.3.4), and a small-data model (Subsection 3.3.5).

3.3.1 *Generic web-tracking model*

The idea of a generic representation of the (third-party) web tracking problem (henceforth: generic web-tracking model) was first proposed by Van Eijk [2011b].²¹¹ Nowadays, other researchers use different terms, e. g., cross-domain tracking [Schneider, Enzmann, & Stopczynski, 2014, pp. 23–24], cookie matching [Mazel, Garnier, & Fukuda, 2017, p. 3], or cookie syncing (see, e. g., Papadopoulos et al. [2018], Acar et al. [2014, p. 682], Englehardt et al. [2015, pp. 6–7], Englehardt and Narayanan [2016, p. 11], Bashir, Arshad, Robertson, and Wilson [2016, pp. 491–493], and Brookman et al. [2017, p. 137]).²¹²

Figure 3.2 (next page) illustrates the generic web-tracking model as an undirected graph. If a tracker (C) receives information from websites (A) and (B) about the same end-user, then (C) may track the end-user across different websites by linking the information. As Barth [2011, RFC 6265] puts this eloquently:²¹³

„Although cookies are not the only mechanism, that servers used to commonly track end-users across HTTP requests, cookies facilitate tracking because they are persistent across user-agent sessions and can be shared between hosts.”

²¹¹ Supra our specialized definition for web tracking (Definition 1.8).

²¹² Example of cookie syncing: I captured the UID '6254484537578825505' being synced between Criteo and AppNexus on the webpage <http://t2tea.com> (16 July 2018): <https://dis.criteo.com/dis/rtb/appnexus/cookieismatch.aspx?appnxsid=6254484537578825505>.

²¹³ Barth is also known for defining the concept Same Origin Policy (SOP) [Barth, 2011, RFC 6265]. The SOP is implemented in modern browsers, e. g., as a security measure to protect e. g., HTTP cookies against the risk of access to the cookie data by JavaScript(s).

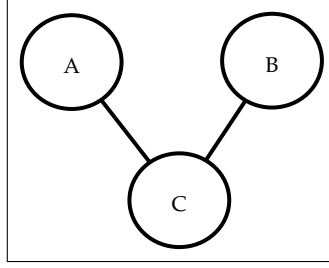


Figure 3.2: Generic web-tracking model [Van Eijk, 2011b].

In *this* thesis we start afresh and formalize the generic web-tracking model. Let us start adopting Definition 3.3 of a labeled property graph as proposed by Riesen, Jiang, and Bunke [2010].

DEFINITION 3.3: „GRAPH. Let L_V and L_E be a finite or infinite label alphabet for nodes and edges, respectively. A graph g is a four-tuple $g = (V, E, \mu, \nu)$, where V is the finite set of nodes, $E \subseteq V \times V$ is the set of edges, $\mu: V \rightarrow L_V$ is the node labeling function, and $\nu: E \rightarrow L_E$ is the edge labeling function.” [Riesen et al., 2010, pp. 219–220].

If we combine Van Eijk [2011b] and Definition 3.3, the formal description of web tracking is as follows.

DEFINITION 3.4: WEB TRACKING. Let us consider a graph denoted by $g = (V, E, \mu, \nu)$ with three nodes (V_A, V_B, V_C), two edges ($E_1: V_A \times V_C, E_2: V_B \times V_C$), and three edge properties ($\mu_A=A, \mu_B=B, \mu_C=C$). If a tracker (C) receives information from websites (A) and (B) about the same end-user, then (C) may track the end-user across different websites by linking the information.

3.3.2 Formalization of graph-mining rules

Van Eijk [2011b] diligently investigated small data to detect their importance for web tracking.²¹⁴ In this subsection we will reflect

²¹⁴ The source code of the specialized HTTP header-analysis software was written in the PERL programming language. The code and the results of 148 different experiments are available in a GitHub repository. URL: <https://github.com/rvaneijk/TDS/> (3 December 2015).

on the findings and formalize the results with respect to Definition 3.4. Formalization of the graph patterns (Figure 3.3) allows us to reason within a system. We discuss (A) illustration and formalization of small data, (B) induced subgraphs and its applications, (C) motif enumeration and counting algorithms, and (D) conclusions on the formalization of graph-mining rules.

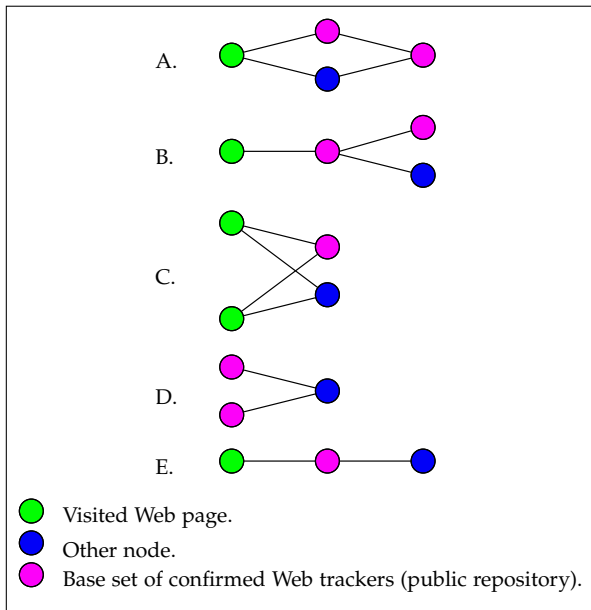


Figure 3.3: Graph mining rules for detecting web tracking in a graph database, e. g., Neo4j. The Cypher Query Language (Cypher) can express these patterns effectively [Van Eijk, 2011b, p. 41].

Van Eijk [2011b] presented nine findings (not reproduced here). On the one hand, his analysis showed two distinct indicators for web tracking:

- (1) the number of filtered nodes as a percentage of the total number of nodes, and
- (2) a cluster coefficient describing the interconnectedness between nodes expressed in the number of links per node.

On the other hand, I formulated five distinct subgraph patterns representing an indication for web tracking. In Figure 3.3 the subgraph patterns are denoted by A, B, C, D, and E.

Van Eijk [2011b] showed that the graph-mining rules represented (1) frequent patterns of web tracking and (2) enabled the detection of missing values through small data. Three colors were used to differentiate between:

- (1) the media website visited (green nodes),
- (2) a web tracking domain not found in a public repository of confirmed web trackers (blue nodes), and
- (3) web-tracking domains found in a public repository of confirmed web trackers (purple nodes).

A: ILLUSTRATION AND FORMALIZATION OF SMALL DATA

To illustrate the concept of *small data* we draw an analogy between DNA and web tracking. Just as the essence of a string of DNA is situated in its small data - i. e., a sequence of nucleotides shaping the right-handed double helix - the essence of the information in the HTTP header fields is situated in its small data.

All subgraph patterns are examples of web-tracking phenomena derived from data analysis from a *host* perspective. The patterns enable the identification of hosts that are unknown for their ability to understand an end-user's browsing interests to target ads and personalize web services accordingly.

Below we formalize each pattern shown in Figure 3.3. Let us start formalizing a subgraph by adopting the definition of a subgraph as proposed by [Riesen et al., 2010] as our Definition 3.5.

DEFINITION 3.5: „SUBGRAPH. Let $g_1 = (V_1, E_1, \mu_1, \nu_1)$ and $g_2 = (V_2, E_2, \mu_2, \nu_2)$ be graphs. Graph g_1 is a subgraph of g_2 , denoted by $g_1 \subseteq g_2$, if

- (1) $V_1 \subseteq V_2$,
- (2) $E_1 \subseteq E_2$,
- (3) $\mu_1(u) = \mu_2(u)$ for all $u \in V_1$, and
- (4) $\nu_1(e) = \nu_2(e)$ for all $e \in E_1$.” [Riesen et al., 2010, p. 220]

The definition of a subgraph enables us to formalize each pattern that is shown in Figure 3.3. The corresponding formalizations of Subgraph A, Subgraph B, Subgraph C, Subgraph D, and Subgraph E are given by Definition 3.6 to Definition 3.10.

DEFINITION 3.6: SUBGRAPH A is defined to be a subgraph denoted by $g = (V, E, \mu, \nu)$ with four nodes (V_A, V_B, V_C, V_D) and four edges ($E_1: V_A \times V_B, E_2: V_B \times V_D, E_3: V_A \times V_C, E_4: V_C \times V_D$), and three distinct node properties ($\mu_A=\text{green}, \mu_B=\text{purple}, \mu_C=\text{blue}, \mu_D=\text{purple}$).

The host visited (V_A) contains two confirmed tracker nodes (V_B, V_D) and one other node (V_C). We consider V_C a tracker node when the interconnectedness between V_A, V_C , and V_D is similar to the interconnectedness between V_A, V_B , and V_D .

DEFINITION 3.7: SUBGRAPH B is defined to be a subgraph denoted by $g = (V, E, \mu, \nu)$ with four nodes (V_A, V_B, V_C, V_D) and three edges ($E_1: V_A \times V_B, E_2: V_B \times V_C, E_3: V_B \times V_D$), and three distinct node properties ($\mu_A=\text{green}, \mu_B=\text{purple}, \mu_C=\text{purple}, \mu_D=\text{blue}$).

The host visited (V_A) contains two confirmed tracker nodes (V_B, V_C) and one other node (V_D). We consider V_D a tracker node when the interconnectedness between V_A, V_B , and V_D is similar to the interconnectedness between V_A, V_B , and V_C .

DEFINITION 3.8: SUBGRAPH C is defined to be a subgraph denoted by $g = (V, E, \mu, \nu)$ with four nodes (V_A, V_B, V_C, V_D) and four edges ($E_1: V_A \times V_C, E_2: V_A \times V_D, E_3: V_B \times V_C, E_4: V_B \times V_D$), and three distinct node properties ($\mu_A=\text{green}, \mu_B=\text{green}, \mu_C=\text{purple}, \mu_D=\text{blue}$).

The hosts visited (V_A and V_B) contain a confirmed tracker node (V_C) and one other node (V_D). We consider V_D a tracker node when the interconnectedness between V_A, V_D , and V_B is similar to the interconnectedness between V_A, V_C , and V_B .

DEFINITION 3.9: SUBGRAPH D is defined to be a subgraph denoted by $g = (V, E, \mu, \nu)$ with three nodes (V_A, V_B, V_C) and two edges ($E_1: V_A \times V_C, E_2: V_B \times V_C$), and two distinct node properties ($\mu_A=\text{purple}, \mu_B=\text{purple}, \mu_C=\text{blue}$).

If the other node (V_C) is connected to two confirmed tracker nodes (V_A and V_B) then V_C can be considered a tracker node as well.

DEFINITION 3.10: SUBGRAPH E is defined to be a subgraph denoted by $g = (V, E, \mu, \nu)$ with three nodes (V_A, V_B, V_C) and two edges ($E_1: V_A \times V_B, E_2: V_B \times V_C$), and three distinct node properties ($\mu_A=\text{green}, \mu_B=\text{purple}, \mu_C=\text{blue}, \mu_D=\text{purple}$).

If the other node (V_C) is interconnected to the visited website (V_A) via a confirmed tracker node (V_B) then V_C can be considered to be a tracker node as well.

B: INDUCED SUBGRAPHS AND ITS APPLICATIONS

We continue for a while our reflection on the findings by Van Eijk [2011b]. Riesen et al. [2010] proposed a definition of an induced subgraph by replacing the second condition in Definition 3.5 as follows.

DEFINITION 3.11: „INDUCED SUBGRAPH. Let $g_1 = (V_1, E_1, \mu_1, \nu_1)$ and $g_2 = (V_2, E_2, \mu_2, \nu_2)$ be graphs. Graph g_1 is a subgraph of g_2 , denoted by $g_1 \subseteq g_2$, if

- (1) $V_1 \subseteq V_2$,
- (2) $E_1 = E_2 \cap V_1 \times V_1$,
- (3) $\mu_1(u) = \mu_2(u)$ for all $u \in V_1$, and
- (4) $\nu_1(e) = \nu_2(e)$ for all $e \in E_1$.” [Riesen et al., 2010, p. 220].

We remark that the concept of an induced subgraph can be applied to our patterns. From Definition 3.11 we can derive two new subgraph concepts. First, it follows that subgraph E is an induced subgraph of graph B. Subgraph E can be obtained from graph B by removing a node.

Second, it follows that subgraph D is an induced subgraph of graph A. This is not obvious at first sight. It is necessary first to determine a blue colored (and therefore uncodified) node as a tracker node, then it becomes clear that subgraph D can be obtained from graph A by removing a node.

C: MOTIF ENUMERATION AND COUNTING ALGORITHMS

Here, we remark that the five subgraphs above are defined manually (Definition 3.6 – Definition 3.10). However, recent developments in network science show that such patterns can be traced

automatically by the application of motif enumeration and counting algorithms (see, e. g., Benson, Gleich, and Leskovec [2016]).²¹⁵

D: CONCLUSIONS ON THE FORMALIZATION OF GRAPH-MINING RULES

In conclusion, our reflection on the formalization of the findings by Van Eijk [2011b] lead us to the following five advantages of adding color to the graph analysis.

- (1) The graph mining rules enable the visualization of web tracking in a force-directed graph.
- (2) The patterns provide a means of detecting missing values, i. e., unknown web trackers.
- (3) The patterns can be formalized.
- (4) The patterns are early examples of subgraph isomorphism applied to the problem of web tracking.²¹⁶
- (5) The patterns enable an understandable graph-based analysis. The rules are queries that can be expressed in a query language such as Neo4j's Cypher Query Language.²¹⁷

3.3.3 *Extended web-tracking model*

Subsequently, Van Eijk [2012] improved the generic web-tracking model. Van Eijk [2011b, p. 32] had already established that a typical (unique) tracking identifier for RTB was at least a ten-digit variable. For an improved model (henceforth: extended web-tracking model) we add four new elements, viz.

- (1) filtering the use of high-entropy identifiers in *any* HTTP header fields, e. g., the ETag header field,
- (2) adding high-entropy as a property to node C (Figure 3.2),
- (3) directing the graph, and
- (4) adding meta data as properties to the edges (e. g., which website sets a UID in a HTTP cookie, or which websites received the UID).

An example of the extended web-tracking model is given in Figure 3.4 (next page). I presented and discussed the example as

²¹⁵ I am indebted to Takes [2018] for this insight.

²¹⁶ Infra Definition 3.17.

²¹⁷ See, e. g., Robinson et al. [2015]; N. White [2014].

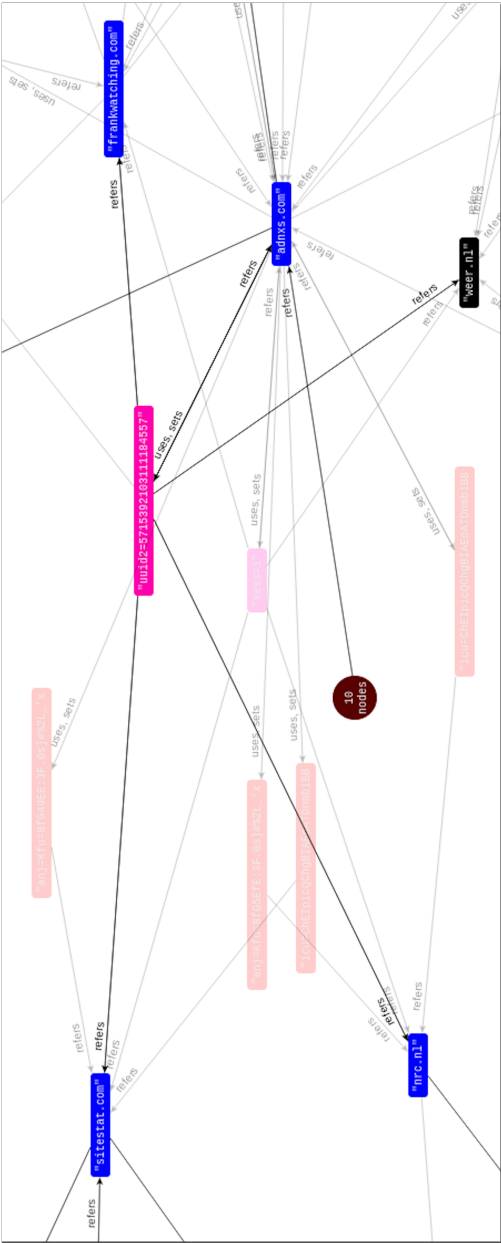


Figure 3.4: The RTB cookie named 'uuid2' with value '5715392183111164557' is set by the domain 'adnxs.com'. An RTB cookie with the same value was connected to the four domains we visited (1) 'nrc.nl', (2) 'weer.nl', (3) 'frankwatching.com', and (4) 'sitestat.com' [Van Eijk, 2012].

an early empirical result of web tracking with UID-cookies at the International Association of Privacy Professionals (IAPP) Europe Data Protection Congress 2012. Figure 3.4 shows a directed property graph.

The graph is compiled with the user friendly Neo4j browser. It enables us to easily query and visualize our research data (Crawl1, Table 3.2). Figure 3.4 contains six nodes in focus ('sitestat.com', 'nrc.nl', 'uuid2', 'frankwatching.com', 'adnxs.com', and 'weer.nl'), one conflated node labeled '10 nodes', and five other (less important) nodes. The Neo4j-browser interface enables us to zoom in on conflated nodes by a mouse click.

At the right of the graph we see domain 'adnxs.com' (AppNexus) as a node in blue color. Although AppNexus is referred to by more than 43 nodes, the focus in our visualization is on the edge between the nodes labeled 'adnxs.com' and 'uuid2=5715392103111184557'. The properties of the directed edge are:

- (A) 'uses',
- (B) 'sets', and
- (C) 'refers'.

The property 'uses' means reading the UID-cookie, whereas the 'sets' stand for setting the UID-cookie in the browser. The property 'refers' denotes a reference to AppNexus in the HTTP header. The central focus on the UID-cookie allows us to see the interconnection by the tracking cookie on the following four domains (blue nodes): (1) 'nrc.nl', (2) 'weer.nl', (3) 'frankwatching.com', and (4) 'sitestat.com'.

3.3.4 High-entropy identifiers

The prevailing question now is: what type of tracking identification do we use for our experiments. Van Eijk [2012] used entropy ([Shannon, 1948]) as a filter for data reduction. The filter uses high-entropy as a criterion for web tracking. As such, it is an example that illustrates our way of modeling metadata.

The main reason to base the experiments on high-entropy cookie IDs was rooted on the idea proposed by Eckersley [2010a], namely that high browser entropy increases the risk identification by a (unique) browser fingerprint (see Section 2.4). Since cookies played (and still do today) a central role in web tracking, the aim of the experiments was to differentiate between tracking cookies

and non-tracking cookies with entropy as a measure. The Shannon entropy is defined as follows.

DEFINITION 3.12: SHANNON ENTROPY is defined to be the Shannon entropy H of a random variable X , with a logarithmic base value j of 2, with outcomes $\{x_1, \dots, x_n\}$, and a probability $P(x_i)$.

The Shannon entropy is given by:

$$H(X) = - \sum_{i=1}^n P(x_i) \log_j P(x_i) \quad (3.1)$$

We define the term high-entropy identifier as follows.

DEFINITION 3.13: HIGH-ENTROPY IDENTIFIER. A Unique Identifier (UID) containing at least 13 bits of (Shannon) entropy, which is sufficient to identify an individual in a group of 5,000 people.

Below we provide (A) three examples of using entropy, (B) an algorithmic description, and (C) observations.

A: THREE EXAMPLES OF USING ENTROPY

The size of the group is consistent with the compliance framework for Do Not Track (DNT) (henceforth: EFF DNT policy) of the Electronic Frontier Foundation (EFF).²¹⁸ The EFF DNT policy contains a normative privacy framework that aims to limit the reconstruction of reading habits or online activity of people using the web. Under the EFF DNT policy, data groups of fewer than 5,000 individuals or devices are considered as identifiable end-users rather than anonymous data. Below we discuss three examples: (A1) a perfect die, (A2) a language code, and (A3) identifiers with multiple elements.

A1: FIRST EXAMPLE, A PERFECT DIE

The Shannon entropy of a single roll with a perfect die is determined by its six sides. Because the chance $P(x_i)$ of the perfect die rolling each number is equal ($\frac{1}{6}$), the entropy H is ~ 2.58 bits of

²¹⁸ See EFF [2015].

binary information to encode the outcome of a single roll with the die.

A2: SECOND EXAMPLE, A LANGUAGE CODE

A second example to illustrate the working of our tracking identification reads as follows. It is language oriented and thus more related to web tracking. Web developers may use a language code in a cookie to store a language preference. ISO [2013, ISO 639-1] defines abbreviations for languages, e. g., 'nl' for the Netherlands. The entropy of language codes is mathematically determined by the number of common languages. The Firefox add-on Privacy-Badger relies on a list of 100 common language codes.²¹⁹ Therefore, the entropy of the listed language codes is ~ 6.64 bits.

A3: THIRD EXAMPLE, IDENTIFIERS WITH MULTIPLE ELEMENTS

A third example sees to identifiers consisting of multiple elements. Only when the probability distributions of each element are independent, the entropy values can be totaled. To understand the dependency of elements, let us consider an example with three elements, e. g., (1) the browser's *local time*, (2) the browser's *time zone*, and (3) the browser's *geolocation* from where the end-user is browsing.

The question here is: can the entropy values of these three elements be totaled? In case the two browser properties *local time* and *time zone* are kept to its default localized values and the *location* where the end-user is browsing from is known through a geolocation lookup of the IP address, the probability distributions of the three browser properties are not independent. The reason for this is that when one knows the *local time* and the *time zone*, knowing the *geolocation* does not increase the entropy.²²⁰

B: AN ALGORITHMIC DESCRIPTION

Van Eijk [2012] applied Shannon's entropy to filter high-entropy identifiers. A threshold was used to differentiate between functional identifiers and tracking identifiers with a high-entropy. Preliminary results of empirical exploratory research were presented

²¹⁹ Cf. Eckersley et al. [2014]. See also Eckersley [2010b; 2015].

²²⁰ Cf. Eckersley [2010b]. See also Koot, Mandjes, Van't Noordende, and De Laat [2011] and Koot [2012].

at the 2012 IAPP Europe Data Protection Congress [Wefers Betsink, Van Eijk, & Wagner, 2012].

Algorithm 3.1 [Van Eijk & Terra, 2012] describes the implementation of Shannon’s entropy.

Algorithm 3.1: Calculating the high-entropy value of a string s .

input : A string s containing an identifier
output: High-entropy value

```

1  $H \leftarrow 0$ ;
2 forall unique characters  $c$  in  $s$  do
3    $P_c \leftarrow \text{s.count}(c)/\text{len}(s)$ ;
4    $H \leftarrow H + P_c \log_2 P_c$ ;
5 return  $-\text{len}(s)H$ ;
```

The high-entropy value of a random string s was calculated as follows. For a string s , each character c is treated as an independent variable. The probability $P(x_i)$ of finding a certain character is given by the number of occurrences of that character in the string divided by the length of the string. The high-entropy value is calculated by multiplying the entropy H with the length of the string. Van Eijk and Terra [2012] implemented the concept in Python.

Determining whether an identifier qualifies as a tracking cookie is a challenge.²²¹ The main issue in a heuristic approach is striking the right balance between (1) false positives (see Definition 3.14) and (2) false negatives (see Definition 3.15).

DEFINITION 3.14: A FALSE POSITIVE is an identifier that qualifies as an heuristically determined high-entropy identifier, but in reality it is not used for web tracking.

For instance, we mention a HTTP cookie containing the version number of a JavaScript file. Reducing the number of false positives requires improving the heuristic approach.

²²¹ We refer to, e.g., Adams et al. [2012], Eckersley et al. [2014], and Reisman, Englehardt, Eubank, Zimmerman, and Narayanan [2014].

DEFINITION 3.15: A **FALSE NEGATIVE** is an identifier that was missed by an heuristic approach. False negatives point to a lack of granularity in the heuristic model.

C: OBSERVATIONS

Asghari, Van Eijk, Englehardt, Narayanan, and Winter [2016] applied Algorithm 3.1 within the framework of their EU-cookie transparency & accountability project (CookieTAP). One of the questions we asked ourselves was: how does entropy as a metric compare to counting third-party cookies? The assumption here was that third-parties do store, e.g., (1) IDs for the end-user and (2) event tracking during session in HTTP cookies.

To answer the question, we set up stateless shallow crawls with OpenWPM [Englehardt et al., 2014].²²² The results of our analysis are shown in Figure 3.5. Their meaning is: it is almost linear (say $\alpha = 40^\circ$ in $y = \alpha x$). This means no difference has been found between the two measures.

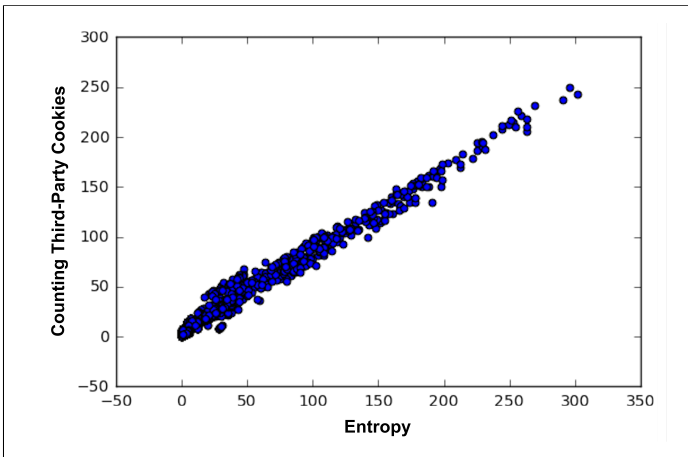


Figure 3.5: Entropy versus Counting Third-Party Cookies (CTPC):
Spearman's correlation coefficient = 0.95758713547808916,
pvalue = 0.0 [Asghari et al., 2016].

²²² We set up stateless shallow crawls with OpenWPM [Englehardt et al., 2014] (name: crawl_20161201.1906_alex10x25_v9). We retained a copy of the crawl data and the Jupyter [Perez, 2001] notebooks in a (private) github repository. URL: <https://github.com/hadiasghari/EUcookies> (2 December 2017).

For closer inspection, we visited a small sample of 140 websites from nine countries, i. e., (1) Austria, (2) Germany, (3) France, (4) United Kingdom, (5) United States of America, (6) Italy, (7) the Netherlands, (8) Poland, and (9) Romania.²²³ To get access from inside a EU-country, we crawled with OpenVPN [Yonan, 2001] endpoints from the IP-address of the Center for Information Technology Policy at Princeton University.²²⁴ Surprisingly, also here we found no difference between the two metrics. The Spearman's correlation coefficient of our dataset is close to one.²²⁵

Van Eijk and Cortesi [2016] also re-investigated the issue and confirmed the result. Moreover, they found:

- (1) entropy (Algorithm 3.1) performed poorly on short cookie values,
- (2) entropy did not match intuition for structured data in longer cookie values, and
- (3) entropy did not distinguish session cookies that did not persist in the browser.

3.3.5 *A small-data model*

After the negative result above, we continued our aim to further improve the extended web-tracking model. Therefore, we propose to formalize web tracking as an exact graph matching problem. This means that we are interested in subgraph isomorphism. The course of this subsection is as follows: (A) subgraph isomorphism, (B) customization, and (C) Ghostery web-tracking classification example.

²²³ The 140 websites were picked from the top-Alexa domain ranking: '.at' (8), '.bz' (1), '.co.uk' (6), '.de' (10), '.fr' (9), '.gov.uk' (1), '.ie' (7), '.it' (11), '.net' (5), '.nl' (7), '.org' (4), '.pl' (16), '.ro' (9), '.ru' (3), '.se' (8), and '.com' (35).

²²⁴ The OpenVPN endpoints were provided by the companies CyberGhost, Privax (HideMyAss), and FoxyProxy.

²²⁵ See also Turcios Rodríguez [2018, pp. 88–89] who re-investigated the issue and measured the correlation between unique cookies names and unique third-party domains ($n = 35,000$ websites, Spearman's correlation coefficient = 0.88792705220847501, $pvalue = 0.0$). Moreover, she measured the correlation between the use of JavaScript and unique third-party domains ($n = 35,000$ websites, Spearman's correlation coefficient = 0.91435850137104757, $pvalue = 0.0$).

A: SUBGRAPH ISOMORPHISM

Below we adopt the definition of graph isomorphism (Definition 3.16) and subgraph isomorphism (Definition 3.17) as proposed by Riesen et al. [2010]. The definitions are as follows.

DEFINITION 3.16: „GRAPH ISOMORPHISM. Let us consider two graphs denoted by $g_1 = (V_1, E_1, \mu_1, \nu_1)$ and $g_2 = (V_2, E_2, \mu_2, \nu_2)$. A graph isomorphism is a *bijective* function $f : V_1 \rightarrow V_2$ satisfying

- (1) $\mu_1(u) = \mu_2(f(u))$ for all nodes $u \in V_1$,
- (2) for each edge $e_1 = (u, v) \in E_1$, there exists an edge $e_2 = (f(u), f(v)) \in E_2$ such that $\nu_1(e_1) = \nu_2(e_2)$, and
- (3) for each edge $e_2 = (u, v) \in E_2$, there exists an edge $e_1 = (f^{-1}(u), f^{-1}(v)) \in E_1$,

such that $\mu_1(e_1) = \nu_1(e_1)$. Two graphs are called isomorphic if there exists an isomorphism between them.” [Riesen et al., 2010, pp. 221–222] (emphasis added)

DEFINITION 3.17: „SUBGRAPH ISOMORPHISM. Let $g_1 = (V_1, E_1, \mu_1, \nu_1)$ and $g_2 = (V_2, E_2, \mu_2, \nu_2)$ be graphs. An *injective* function $f : V_1 \rightarrow V_2$ from g_1 to g_2 is a subgraph isomorphism if there exists a subgraph $g \subseteq g_2$ such that f is a graph isomorphism between g_1 and g .” [Riesen et al., 2010, p. 223] (emphasis added)

Bijection is a mathematical term that describes a one-to-one correspondence, or the exact matching of two graphs. The term injection describes a one-to-one function that preserves distinctness, or exact matching of a subgraph.

B: CUSTOMIZATION

Customized graph data models (henceforth: small-data models) are particularly suitable when it comes to solving graph matching problems. The key objective for modeling the small data is to include all key properties needed for subgraph isomorphism. The properties correspond with specific properties (metadata) of the research data. Small-data modeling by feature extraction from the HTTP headers enables encoding of research questions in a

way that feels like natural speech. The metadata in the data flow allow us to model the small data. We provide a telling example: (B1) an example of customization, including (B2) an algorithmic background of the example.

B1: AN EXAMPLE OF CUSTOMIZATION

To illustrate small-data modeling, we turn to event-level tracking for email. The example stems from a personal account. Recently I signed up for a trial period for cloud-based transcription software.²²⁶ At the end of the trial period I received a notification email containing a friendly reminder to purchase a software license.²²⁷ The email reminder contained a hyperlink with a HTTP redirect.²²⁸ Instead of taking me straight to the login page for my account, the login page loaded after having been redirected from a URI recording the click event.

URI redirection is a common HTTP mechanism to instruct the browser to load a resource that is located somewhere else (cf. Fielding and Reschke [2014b, RFC 7231, Section 6.4]). The HTTP response status code 302 indicates URI redirection. HTTP redirects can also be used to uniquely identify end-users, a tracking technique known as ‘301 moved permanent redirect tracking’.²²⁹

A small-data model for event-level tracking for email is shown in Figure 3.6. The aim of the small-data model is to investigate, e. g., the following question: which URIs requested a resource from a third party and were redirected to a resource from another third-party?

The small-data model illustrates how the phenomenon of URI redirection could be encoded in a directed graph model. The research data stems from the HTTP headers.

The small-data model (Figure 3.6) has edge labels and node labels. The labels ‘REQUEST’ (encoding of the HTTP GET request) ,

226 URL: <http://transcribe.wreally.com/login> (26 December 2015).

227 „You can do so [purchasing a license] by logging into your Transcribe account and visiting your account page”.

228 The URI was as follows URL: [http://emails.transcribe.wreally.com/track/click/30097249/transcribe.wreally.com?p=eyJzIjo1VDIwc1J\(...\).SJ9](http://emails.transcribe.wreally.com/track/click/30097249/transcribe.wreally.com?p=eyJzIjo1VDIwc1J(...).SJ9) (26 December 2015) (shortened).

229 Mozilla [2018b]: „301 moved permanent redirect tracking: if you visit a site, it might load a resource that has a 301 redirect. The resource can redirect you to URL that is unique to you. Then, the next time you see the original resource, your browser will load the unique URL from the cache, and fetch the resource from there. Making you uniquely identified.”

'REDIRECT' (encoding of the HTTP 302 redirect), and 'NEXT' (encoding of two consecutive URI visits) are edge properties. Above we have modeled HTTP redirection in a graph-based object. It enables us to formalize the subgraph with the following definition.

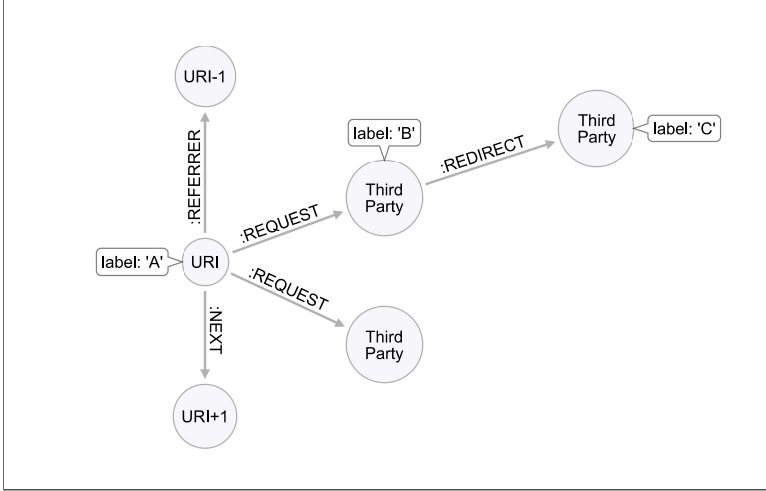


Figure 3.6: Example of a small-data model to investigate HTTP redirection through subgraph isomorphism.

DEFINITION 3.18: HTTP REDIRECTION. Let us consider a directed subgraph denoted by $g = (V, E, \mu, \nu)$ with six nodes ($V_1, V_2, V_3, V_4, V_5, V_6$), five edges ($E_1: V_1 \leftarrow V_2, E_2: V_2 \rightarrow V_3, E_3: V_2 \rightarrow V_4, E_4: V_2 \rightarrow V_5, E_5: V_4 \rightarrow V_6$), six distinct node properties ($\mu_1 = \text{URI-1}, \mu_2 = \text{URI}, \mu_3 = \text{URI+1}, \mu_4 = \text{'Third Party'}, \mu_5 = \text{'Third Party'}, \mu_6 = \text{'Third Party'}$), and five distinct edge properties ($\nu_1 = \text{'REFERRER'}, \nu_2 = \text{'NEXT'}, \nu_3 = \text{'REQUEST'}, \nu_4 = \text{'REQUEST'}, \nu_5 = \text{'REDIRECT'}$)

HTTP redirection is defined to be an induced subgraph g' of g , if g' consists of the three nodes (V_2, V_4, V_6) and the two edges (E_3, E_5).

B2: ALGORITHMIC BACKGROUND OF THE CUSTOMIZATION EXAMPLE

To better understand the practicality of encoding questions in a small-data model we look at the place where to look in the research data. We restrict the example (Listing 3.3) to URI redirection. The induced subgraph we are interested in consists of the nodes labeled A, B, and C in Figure 3.6. The (simplified) HTTP request header is given in Listing 3.3.

Listing 3.3: HTTP request header with a GET request to a web resource.

```

1 GET /index.html HTTP/1.1
2 Host: www.domain-a.nl
3 ...

```

In response to the HTTP request above, the server responds with a HTTP 302 status code. The response is given below in Listing 3.4.

Listing 3.4: HTTP response header containing a HTTP 302 status code and the HTTP Location header field containing the redirect URI.

```

1 HTTP/1.1 302 Moved Temporarily
2 Location: http://www.third-party.nl
3 ...

```

The HTTP status code, and the domain names of the URIs - 'uri-1.nl' (Listing 3.3, r. 2) and 'third-party.nl' (Listing 3.4, r. 2) - are node properties of their respective nodes. A Cypher query encoding the example question could be based on the isomorphic pattern, given in Listing 3.5.

Listing 3.5: Subgraph isomorphism: Matching pattern for a Cypher query.

```

1 (URI) -[REQUEST]->(Third Party) -[REDIRECT]->(Third Party)

```

Of course, small-data models for WPM research do not have to be restricted to HTTP-header information. In summary, what we intended to show is the possibility that metadata in the data flow

allows us to model the small data. We believe that this point has been made.

C: GHOSTERY WEB-TRACKING CLASSIFICATION EXAMPLE

A topic that is somewhat related with the small-data model is Ghostery [Pierce, 2010]. Below we show a tracking protection technology based on the generic web-tracking model.²³⁰

A relevant development for our research is the possibility to block web traffic. In the example below we demonstrate the limitations of blocking web tracking when using the well-known browser extension Ghostery [Pierce, 2010]. Ghostery is (1) a transparency tool for web tracking, but (2) the extension gives end-users (limited) control over blocking tracking elements in a webpage. So, Ghostery blocks web-tracking elements with a blacklist approach. The effectiveness of blacklist blocking has been studied by WPM scholars, see, e. g., Mayer and Mitchell [2012, p. 12].

Below we briefly discuss (C1) the modern tracking-protection technologies, and (C2) the generic web-tracking model with node classification.

C1: TRACKING PROTECTION TECHNOLOGY

Nowadays browser manufacturers endow their browsers with built-in tracking protection. Of course, end-users are also allowed to configure the browser with (experimental) tracking extensions. Although not enabled by default, shipping a browser with tracking protection technology is a major step in terms of end-user empowerment, since then the underlying privacy question (Section 1.2) is at stake. We can subdivide the underlying technology into three types:

- (1*) host based,
- (2*) regular expressions, and
- (3*) tracking classification.

Below we study this protection technology in some details. For a proper understanding, we provide brief descriptions for each technology.

²³⁰ Generic web-tracking model (Figure 3.2). For the examples I drew inspiration from N. White [2014] who demonstrated the use of RNeo4j.

1*: **HOST BASED** Microsoft has been a proponent of host-based tracking-protection lists. Although Microsoft does not maintain a list itself, end-users can configure the Internet Explorer with host based lists available online.²³¹ Mozilla ships Firefox with a tracking protection list provided by its partner Disconnect [Kennish et al., 2013].²³²

2*: **REGULAR EXPRESSION** is a programming method for locating specific character strings embedded in character text [Thompson, 1968]. A well-known list based on regular expressions is Easylist [Petnel, 2005]. *Ghostery* also uses this tracking protection technology.

3*: **TRACKING CLASSIFICATION** Apple's Intelligent Tracking Prevention (ITP) is a state of the art technology based on machine learning. Apple ships its Safari browser with ITP (iOS 11). Apple's algorithmic approach is based on three indicators for tracking (cf. Wilander [2017]).²³³

- (1) a subresource on the number of unique domains, e. g., a tag (see Subsection 4.3.8),
- (2) a subframe on the number of unique domains, e. g., a Google SafeFrame, and
- (3) the number of unique domains redirected to (see Figure 3.6).²³⁴

C2: GENERIC WEB-TRACKING MODEL WITH NODE CLASSIFICATION

In Figure 3.7 and Figure 3.8 (p. 126) we demonstrate the practicality of small-data modeling with node classification.

231 URL: [https://msdn.microsoft.com/en-us/library/hh273399\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/hh273399(v=vs.85).aspx) (27 November 2017).

232 See, URL: <https://support.mozilla.org/en-US/kb/tracking-protection> (27 November 2017). The latest version of the blacklist is available online. URL: <https://github.com/disconnectme/disconnect-tracking-protection/blob/master/services.json> (27 November 2017).

233 The original quotation is as follows: „A machine learning model is used to classify which top privately-controlled domains have the ability to track the user cross-site, based on the collected statistics. Out of the various statistics collected, three vectors turned out to have strong signal for classification based on current tracking practices: [1] subresource under number of unique domains, [2] sub frame under

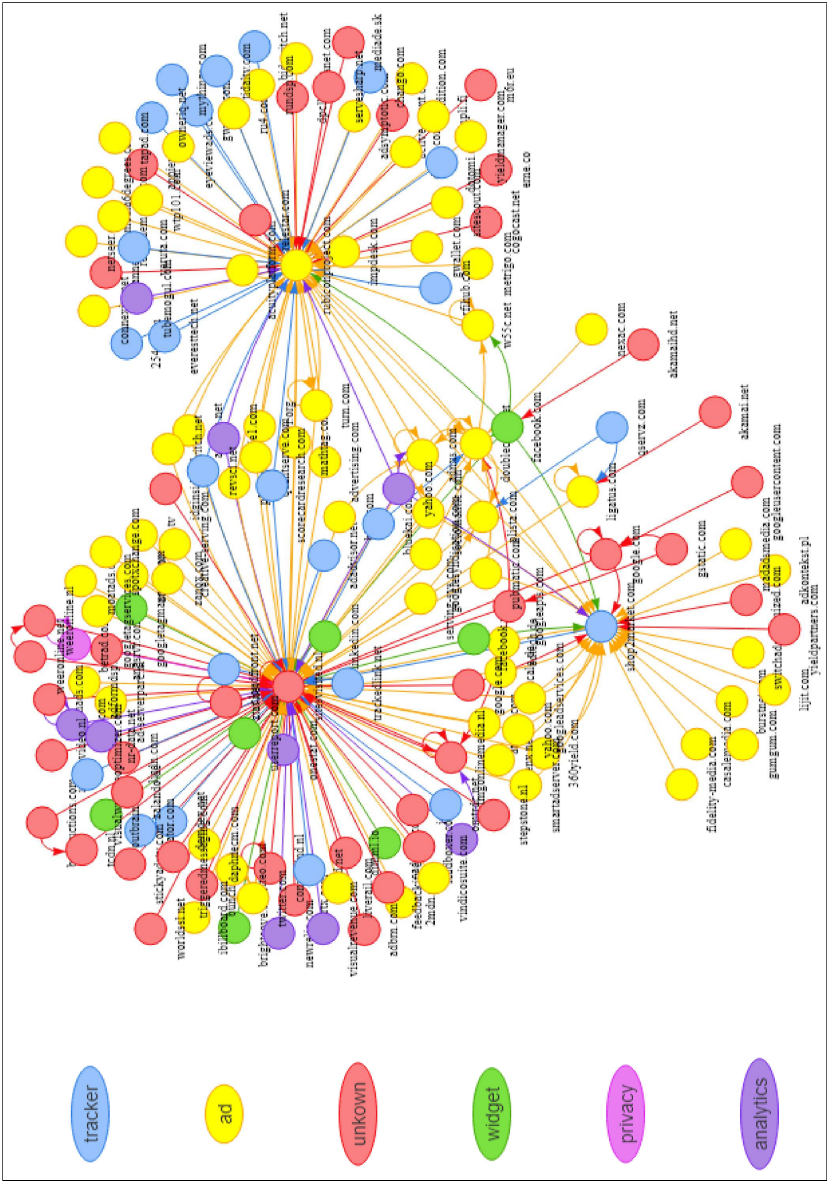


Figure 3.7: Generic web-tracking model with Ghostery-node classification (N=158 nodes and E=228 edges).

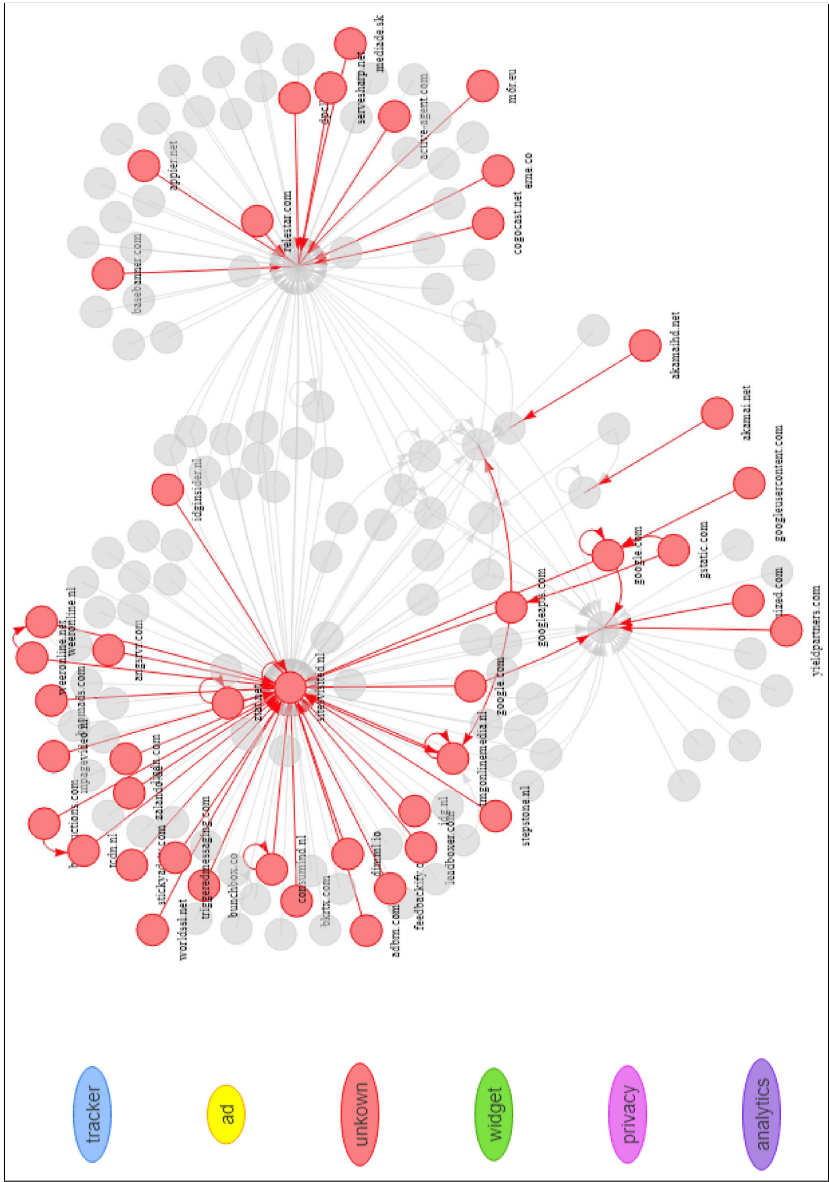


Figure 3.8: Ghostery-node classification: missing values.

We analyzed Ghostery's tracking classification. The Ghostery browser extension contains metadata to classify web tracking.²³⁵ The result of our analysis is visualized in an undirected property graph. Figure 3.7 shows all categories whereas Figure 3.8 highlights the missing values (nodes in red color denoted by the category 'unknown').

The graph represents data from manually visiting four Dutch websites.²³⁶ We applied the generic web-tracking model to the data with the R-package *visNetwork* [Thieurmél, 2016].²³⁷

To make the graph more meaningful, we combined the individual nodes of the four sites visited into one node ('sitesvisited.nl').²³⁸ The graph consists of 158 nodes. The Ghostery metadata enables us to add a color as a property for the tracking category to the nodes.

The six categories denoted by Ghostery's metadata are:

- (1) tracker (24 nodes),
- (2) ad (72 nodes),
- (3) unknown (44 nodes),²³⁹
- (4) widget (8 nodes),²⁴⁰
- (5) privacy (1 node),
- (6) analytics (9 nodes).

In summary, Ghostery is known for providing transparency in web tracking and giving its end-users more control over their (private) data when they browse the web. Our experiment shows that 44 nodes (27 %) still received (personal) data because they were

number of unique domains, and [3] number of unique domains redirected to." [Wilander, 2017]

234 I am indebted to O'Neill [2017] who confirmed Apple's algorithmic approach to Apple's ITP.

235 The metadata was released by its owner Cliqz under an open source license in March 2018. URL: <https://github.com/ghostery/ghostery-extension/tree/master/databases> (9 March 2018).

236 The websites visited are: (1) webwereld.nl, (2) www.debienkorf.nl, (3) www.zalando.nl, and (4) www.telegraaf.nl. We retained the data with Mitmproxy (filename: 150324-bienkorf-webwereld-test.mitm).

237 We analyzed the data in R [R Core Team, 2014]. We shortened the URL to the domain name and added it as a property to the nodes. We used the referrer field (HTTP request header) and location field (HTTP response header) as information for the edges. The *visNetwork* [Thieurmél, 2016] package comes with a feature to create a web application which simplifies further analysis of the graph.

238 Infra our definition for graph refactoring (see Definition 4.13).

239 We colored the central node 'sitesvisited.nl' red.

240 For example a Facebook-like button

classified as missing values ('unknown') by the browser extension. In general, the classification problem is an illustration of the shortcomings of a blacklist approach. We mark that a blacklist approach to prevent web tracking will not be 100% complete.²⁴¹ We refer to, e.g., Englehardt et al. [2018, pp. 9–10], Merzdovnik, Huber, Buhov, Nikiforakis, Neuner, Schmiedecker, and Weippl [2017], Traverso, Trevisan, Giannantoni, Mellia, and Metwalley [2017], and Mazel et al. [2017] (who proposed a reliable methodology for privacy protection comparison, including blacklist approaches, and extensively compared a wide set of privacy protection techniques).

We remark that Ghostery acknowledged the gap in tracking protection (see Konrad [2017]).²⁴² The company improved their product's tracking-protection capabilities with an algorithmic approach to rewriting UIDs as a measure to reduce the privacy risk of tracking end-user behavior.²⁴³

To better understand the missing values, we will further investigate the RTB landscape with its multiple actors and their interrelationships in Chapter 4. There we show two intricacies which we handel with two M-steps (see Subsection 4.4.3). Finally, we remark that Ghostery acknowledged the gap in tracking protection [Konrad, 2017].²⁴⁴ The company improved their product's tracking-protection capabilities with an algorithmic approach to rewriting UIDs as a measure to reduce the privacy risk of tracking end-user behavior.²⁴⁵

²⁴¹ The same applies to, e.g., Mozilla's (always on) blacklist approach for tracking protection [Mozilla, 2018a].

²⁴² The full quotation is as follows: „But as tracking companies constantly update existing trackers and add new ones, it would be impossible to detect and block all trackers on the internet.”

²⁴³ The technology was developed by Cliqz: „Ghostery 8's enhanced privacy protection detects and overwrites any uniquely identifying data points being passed to a third party - even trackers that aren't caught by Ghostery's comprehensive blacklist” [Konrad, 2017].

²⁴⁴ The full quotation is as follows: „But as tracking companies constantly update existing trackers and add new ones, it would be impossible to detect and block all trackers on the internet.”

²⁴⁵ The technology was developed by Cliqz: „Ghostery 8's enhanced privacy protection detects and overwrites any uniquely identifying data points being passed to a third party - even trackers that aren't caught by Ghostery's comprehensive blacklist” [Konrad, 2017].

3.4 CHAPTER CONCLUSIONS

In this chapter we presented our GBMA by investigating three parts: data collection, data reduction, and data modeling.

In Part 1, data collection, we proposed a new 3-step procedure (C-steps) of data flow collection (see Subsection 3.1.3). In Part 2, data reduction, we again proposed a 3-step procedure (R-steps) of reducing the research data (see Section 3.2). In Part 3, data modeling, we proposed a small-data model, in which the metadata are stored (see Section 3.3).

Here, we remark that an analysis of the strengths and weaknesses of the GBMA is (to a large extent) presented in Section 5.2. Furthermore, we remark that small-data modeling is a new development in comparison with (1) the generic WPM framework [Englehardt et al., 2014] and (2) our extended web-tracking model [Van Eijk, 2012]. By modeling the context with specific small-data elements we will later address RQ2 (see M-step 1 and M-step 2, Subsection 4.4.3). At this moment we will focus on RQ1.

3.5 AN ANSWER TO RESEARCH QUESTION 1

We are now able to answer RQ1: how do we move from data collection to graph analysis? To answer this question we presented a Graph-Based Methodological Approach. Through the GBMA we showed that small-data models can be based on metadata.

Since the HTTP header contains valuable information needed to build small-data models for web tracking, we thoroughly examined data collection (Section 3.1) and data reduction (Section 3.2). Building models is only possible with sufficient knowledge of data modeling (Section 3.3).

Our findings were extensively explained and thoroughly tested. We arrived at the following path that straightforwardly goes from data collection to graph analysis in three well defined parts.

FINDING 1: The properties of the nodes and edges in the small-data model correspond with metadata in the research data. For instance, the edges in the small-data model may be directed or weighted, to express specific properties of the metadata.

FINDING 2: The main improvement of the capabilities for analysis is the fact that a small-data model allows us to apply a variety of algorithms to the data.

FINDING 3: Such a varied algorithmic approach allows us to deepen our understanding of what actually happens when we visit websites containing RTB advertisements. This is the answer to RQ1.

The new and deep insight into the intricacies of web tracking is important for science, but also for society, since it substitutes a warning to privacy protections: be aware of the power of modern technologies.