TOWARDS BETTER WEB MEASUREMENTS BY MITIGATING IMPACTING FACTORS

by

Jorgos Korres

in partial fulfillment of the requirements for the degree of

Master of Science
in Software Engineering

at the Open University, faculty of Science
Master Software Engineering
to be defended publicly on Tuesday 18 January at 3:00 PM.

Student number:	851378513	
Course code:	IM9906	
Thesis committee:	dhr. dr. Fabian van den Broek (chairman),	Open University
	dhr. dr. ir. Hugo Jonker (supervisor),	Open University
	M.Sc. Benjamin Krumnow (supervisor),	TH Köln

ACKNOWLEDGEMENTS

First and foremost, we want to thank our research supervisor, Hugo Jonker, and his assistants Benjamin Krumnow and Fabian Van Den Broek. With-out their assistance and dedicated involvement in every step throughout the process, this research would have never been accomplished.

We also place on record, our sense of gratitude to one and all, who directly or indirectly,have lent their hand in this venture

CONTENTS

List of Figures

LIST OF TABLES

SUMMARY

Various factors that arise from the scale of web measurements and the complexity of the internet impact web measurements in today's studies. Improving web measurements allow researchers to increase the quality of their collected data and conclusions about them. In general, privacy studies that focus on user tracking identify and handle only a few of these factors. Thus, their research does not grasp the entire scraping environment's complexity, and their results are possibly poisoned. This poisoning can have an impact on their concluding statements.

We performed an extensive literature study structured around a generic scraping model to identify a complete set of impacting factors and related mitigations. We then developed a taxonomy driven by the decomposition of the scraping model and the categorisation of the impacting factors. We eventually identified thirty-one impacting factors and corresponding mitigations that we present in our taxonomy. Further analysis of our taxonomy revealed relations between its categories and the mitigations we can use to search for still unknown factors. A final review of quantification methods to measure the impact of the factors and mitigation resulted in a recommendation for the use of violin plots, which combines the advantages of the evaluated approaches.

Our research results allow other researchers to step out of the box with its limited view to become aware of the wide and complex web scraping environment. Applying this knowledge to the situation in their research can, first of all, reveal new impacting factors and hidden problems, for which they can more easily search for a mitigation technique. Furthermore, they can learn and analyse the relationships between their identified impacting factors using our taxonomy. Above that, if researchers are uncertain about the true impact of any factor or mitigation on their measurements, they can use our quantification technique, which is generally applicable to most of our impacting factors, to compare web scraping results and make an in-depth judgement of its impact.

1

INTRODUCTION

Web bots are an exciting tool for privacy researchers. Their ability to research privacy-related issues on huge amounts of websites comes in handy because privacy on the internet has become a real concern that reaches every user. Examples of privacy concerns are the leaking of personal information through cookies in the URL, and in the meantime, we all have heard of the large-scale tracking practices by Facebook. International organisations, such as the EU, have acknowledged this and tried to fill this gap in the field of internet privacy. Member states, for example, adopted the General Data Protection Regulation (GDPR)[1]. Also, the further development of web bots can help in this field by, for example, improving the possibilities to research privacy-related topics.

Tracking has evolved into a significant cause of privacy violations. Recent studies have researched different tracking related topics in the context of internet measurement. This type of research involves visiting numerous websites, even up to 1M. Web scraping, the process to automatically visit all these sites, requires specialised automated tools, called a web scraper, or more generally, a web bot. An example of a study that performs web scraping to study tracking using a web bot is the study of Englehardt and Narayanan [EN16]. The study made use of a newly developed scraping tool called OpenWPM.

Web scraping and privacy measurements seem like the perfect match. However, the modern internet is very complex. Many factors come into play to accomplish a successful web scrape because many things can go wrong too. A successful web scrape, then, is a web scrape that ends without errors or delays and which furthermore does not hook us to permanent countermeasures. Web bot detection and individualisation are examples of factors affecting research results. Some of these factors, such as the impact of location and cloaking, were the subject of whole separate research studies. All studies that measure privacy-related subjects, acknowledge that these factors can impact their scraping results. This acknowledgement translates itself into a description of the confounding factors and how they tried to mitigate them. A confounding factor in research is a different factor than the main variable of the research, but one that can also influence the research results. In the context of our research and the studies that we analysed, they have a more specific meaning. Here, the confounding factors refer to the cause that different web scrapes to the same website result in a different website response. We noted that most studies mention

[1] https://ec.europa.eu/info/law/law-topic/data-protection/data-protection-eu_en

only a handful of these confounding factors (or impacting factors), and, above that, different studies mention different kinds of factors. We now used the term impacting factors, which we will use throughout our research. We defined this new term because we want to detach the common confounding variables from their research and generalise their meaning. Because of these differences in identified impacting factors between the studies, some studies may fail to acknowledge and mitigate some impacting factors that can affect their research results.

So we miss a substantial overview of all the factors that may impact web measurements that employ web scraping techniques and their associated mitigations. Our research wants to go a few steps further to fill this knowledge gap by identifying all the possible factors involved and the most effective mitigations. A complete set of impacting factors will allow us to extend our limited view of the web scraping environment. This wider overview has the advantage that a researcher can analyse his approach and identify any unknown problems. Problems are there to be resolved. With our search and analysis of mitigations, we may also provide a solution to those problems.

The amount of impacting factors would be too chaotic to analyse altogether. We want to bring order to this chaos by categorising all impacting factors and mitigations. We can use a taxonomy for this. A taxonomy will allow us to get a complete overview of all impacting factors to understand the scraping environment better. Furthermore, it will also allow us to analyse the relationships between the impacting factors, find new categories, and derive additional knowledge from them. Perhaps we could even find new impacting factors. A taxonomy is certainly a tool that supports such tasks.

Getting to know your problem and solving them is one thing, but what really matters for your research is whether they impact their research measurements. We certainly are not interested in trivial issues. To assess this impact, we need to measure and quantify the impact on your research results. A quantification method to adequately compare and analyse the results would come in really handy. Because of this, we present a quantification method specifically tailored to compare scraping results.

To acquire the knowledge to accomplish our research, we mainly rely on the results of an extensive structured literature study. Because of the fast-paced advances of the internet and web scraping, we specifically limit the scope to literature from the last six years. Furthermore, we limit the studies' topics to studies that perform web privacy measurements that focus on user tracking-related topics. Privacy-related studies that employ web scraping are just a subset of the total scope of web scraping related literature. However, we noted that general studies that employ web scraping techniques treat web scraping techniques and validate their results on a higher, less technical level, while the privacy-related studies that we analysed treat the subject on a more specialised and higher technological level. Besides that, privacy is a sensitive subject to users and website owners, who prefer to hide any questionable practices. Such behaviour forces privacy researchers to spend a lot of attention on carefully analysing differences in the website responses and validating their findings.

Differences between web scraping-related studies affect the impacting factors that we can identify. We, first of all, have those factors specifically related to the study design. Next, there also exist factors related to user tracking, such as fingerprinting. We aimed to find a complete list of impacting factors, but we based our search on a single generic scraping model that follows the state-of-the-art in web scraping studies.

We organised this thesis in the following way. After an overview of the privacy and tracking related background information, we present the related work that we consulted for our research. Next, we briefly discuss our used methodology. Afterwards, we spent four chapters on the results of our research. We start by presenting the impacting factors that we identified. We then process all this information into a taxonomy. Next, we discuss the mitigations for these impacting factors, and we also give some additional attention to any impacting factors that we may have missed. We finally analyse our impacting factors and mitigations a bit further by determining how we can quantify their impact on the researched studies their measurements. We conclude this thesis with a discussion on the results, a future work chapter, and a conclusion.

We want to reach our research objective by answering the following four research questions:

RQ1. Which impacting factors are known?

RQ2. Which mitigations for these impacting factors are known in literature, and how can we handle unknown factors?

RQ3. How can we categorise the found impacting factors?

RQ4. Which quantification methods to measure the effect of impacting factors and mitigations on measurements exist, and what recommendation can we make based on their evaluation?

Altogether, our research methods will allow us to contribute to the state-of-the-art web measurements that use web bots to study privacy-related problems. In this thesis, we achieved three major contributions:

1. Our extensive structured literature study resulted in the identification and mitigation of thirty-one impacting factors.

2. We developed a taxonomy of our impacting factors based on the analysis and decomposition of the scraping model and categorisation of the identified factors.

3. We developed a quantification method to measure the impact of a subset of our impacting factors and mitigations, based on violin graphs.

2

BACKGROUND

We carried out our research within the wide context of web privacy measurements. To give you an impression of this context and introduce the core concepts of the studies that we consulted, we present you this background section.

This chapter will first introduce the privacy context regarding user tracking to give you the necessary background knowledge. Afterwards, we will describe the user tracking techniques, such as third-party tracking, cookies, invisible pixels and fingerprinting. Next, we will give a short introduction to scraping and scraping studies. Finally, we will cover web bot detection and its identification by fingerprinting and behavioural web bot detection.

2.1. PRIVACY

In recent years, developers have been making a great effort to develop privacy-preserving technologies—for instance, tracker blockers, and VPN proxies. People should indeed worry about their privacy because, besides the many issues, companies treat your personal data as a commodity [PKM18] that is bought and sold on an ad hoc basis.

With privacy, we generally think of hiding what we write to each other and the personal information that we store. But in our research, we are looking more at the concept of privacy related to user tracking. Private information, in this case, includes the websites that you visited and your browsing behaviour (what you watch, where you click on). Indeed, one of the privacy concerns on the internet, according to Lerner et al. [LSKR16], is that tracking companies build lists of websites that users have browsed. Your browsing history can reveal a lot of personal information about you [MM12], such as, for example, location, interests, purchases, sexual orientation, and medical conditions.

Users seem to have serious concerns about this. According to some user surveys [MM12], users have consistently shown opposition to the practice by third parties to collect and use browsing activities. They, for example, would not want any advertising based on tracking. Governments have translated these user concerns to GDPR regulations. However, Mayer and Mitchell note that policy views on third-party web tracking vary substantially, and there are many points of disagreement on specifics.

The oldest culprit of our concerns are cookies [RL18], of which third-party cookies have been deteriorating privacy on the web since the early nineties. But there are more technologies that make things worse. Lerner et al. [LSKR16] also identified other forms of tracking, such as HTML5 Local Storage and browser or machine fingerprinting, which use JS

fingerprinting-related APIs. Developing privacy-preserving tracking technologies is an ongoing work where most techniques still suffer from usability problems. At the same time, we see increasing use of persistent tracking mechanisms, such as Evercookies. These persistent tracking mechanisms have the potential to circumvent the user's tracking preferences [AEE+14]. They furthermore are hard to discover and resilient to remove.

2.2. User Tracking

With tracking, we can think of all kinds of entities collecting and using people's private information as they surf the web. Personal data is a valuable asset for any company [PKM18], and it is the fuel of the free internet. The free internet includes services, such as free mail, search engines and free cloud storage. Tracking makes the free internet possible because sites make money on showing targeted ads. Targeted ads, in their turn, target user profiles that are based on the tracking information that websites collect and sell [VNBJ14]. Companies attract users, collect information about them, and finally monetise on this information.

Researchers are studying tracking for quite some time now. The first measurement studies on tracking appeared around the year 2005. Lerner et al. [LSKR16] later studied tracking (third-party cookie tracking) over a more extended period, from 1996 to 2016. They concluded that the situation is deteriorating over this period. Their measurements show that third-party tracking has increased in prevalence and complexity and that, further on, different forms of tracking behaviour has emerged. Other state-of-the-art studies that measure tracking mostly rely on filter lists, such as EasyList, EasyPrivacy, and Disconnect [FBLS20]. These lists are blacklists of known advertising and tracking domains, and are used by popular browser plugins that block trackers.

Third parties

In web privacy measurements, we generally make a distinction between first parties and third parties. If we take an example website name (or hostname) www.example.co.uk, then we say that this website has the Top Level Domain plus one (TLD + 1) of example.com. The first party is then the website you are trying to visit, and the third party is someone with another TLD + 1 than the one you are visiting.

Many websites embed third parties, mainly without the visitors being aware of them. They are primarily involved in the businesses of advertising, analytics, social networking [MM12], and further on, also in the businesses of content providers, frontend services, and hosting platforms [TJM15]. Websites can embed them through images, multimedia content, fonts, JavaScript libraries, style sheets etc. These web resources are commonly hosted on third-party domains and delivered via content delivery networks [RL18].

Third parties provide these cool services, but, as we have said, the free internet is not free for nothing. Third parties are in a great position for tracking, and above that, there are also positive reasons for such tracking [TJM15], for example, fraud prevention and suggesting related content. Lerner et al. [LSKR16] defines third-party tracking as:

"The practice by which third parties, such as advertisers, social media widgets, and website analytic engines - embedded in first-party websites that users visit directly - re-identify users across domains as they browse the web."

Identifying users across domains is also called cross-domain tracking. An example ex-

plained by Torres et al. shows how Facebook can track users across the web. A website includes a code that instructs the browser to contact the Facebook servers to download their "Like" button. While doing this, the browser sends Facebook which URL triggered this request through the HTTP Referer field. If this user visits many websites that embed this Facebook "Like" button, then Facebook can reconstruct the browsing history (or part of it) of this user.

This cross-domain tracking by third parties comes at a severe privacy cost. It negatively impacts the user's privacy without being of much benefit to him [TJM15]. Furthermore, it allows unauthorised and unrelated third parties to retrieve information from the first party websites and even perform actions on them [RL18].

COOKIES

Cookies, initially a technique to maintain state at the client side, quickly proved their use for tracking purposes. Cookies work by setting a unique identifier [PKM18] that websites use to identify users across sessions and domains. In this case, the mechanism works as follows [VNBJ14]: the webserver stores a small amount of data (the identifier) on the computer of the website visitor. The client later sends back this small amount of data when making subsequent requests to the website.

There are two different types of cookies [PKM18, VNBJ14]. We have first-party cookies, which the website that the user visits sets. Websites use these to maintain state and to track users when they repeatedly visit the website. We also have third-party cookies given by third-party servers that provide their content to a first-party website. Trackers (or third parties) use them to track users across websites. Over some time, they can build a user profile for each user that they can monetise on.

Figure 2.1 illustrates the basic cookie mechanism. In this case, a user uses a Chrome browser first to visit theonion.com and later cnn.com. Both websites make an additional request to tracker.com. If the user makes a request to tracker.com for the first time, the tracker will embed a cookie with a unique identifier in the user's browser. Now, the cookie was set, and the user's browser sends the cookie along with the request. This action happens for both websites. The tracker has now learned that the user visited both theonion.com and cnn.com. In case that both websites would embed the Facebook "Like" button, then both websites would learn that the user visited the two websites (theonion.com and cnn.com).

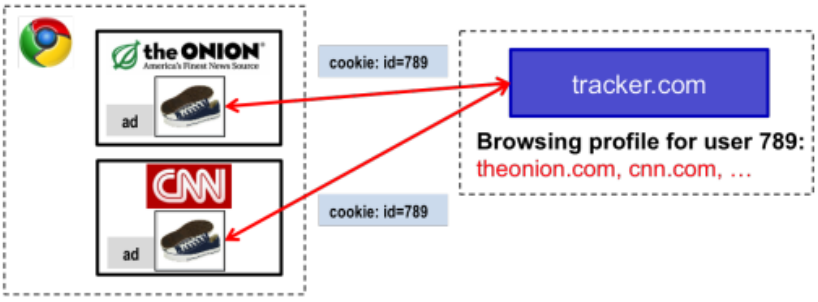

Figure 2.1: Basic Cookie mechanism where both websites make an additional request to tracker.com and include the user's identifier, Source: [LSKR16]

Cookie Synchronisation: (CSync) If cookies were already a problem for your privacy, then cookie synchronisation will be a huge problem. Basically, cookie synchronisation is about trackers who both set their own cookies and later share the user identifiers. So cookie synchronisation facilitates an information-sharing channel [PKM18], where third parties merge the user data that they own in their back-end databases.

This sharing all happens, hidden from the website visitors, because the communication is not directly observable. It further on also allows to bypass the Same Origin Policy (SOP) [AEE+14]. This policy says that a website can only collect cookie data from a domain or sub-domain that sets the cookie, so one domain cannot access the cookie set by another domain [FBLS20]. The privacy implications are that these third parties can reconstruct a larger fraction of a user's browsing history and patterns [AEE+14] and that this is possible even if there is no direct collaboration between the website and the third party [PKM18]. The results of the study by Englehardt and Narayanan [EN16] show that these server-to-server user data merges are taking place at a massive scale.

Figure 2.2 shows a basic scenario of cookie synchronisation. Starting from the top left part. A user visits "website1", which includes a JavaScript script from the third-party tracker.com. This tracker sends in response a cookie with a unique identifier; in this case, the identifier is user123. Next, the user visits a second website, "website2", which includes an embedded banner from a third-party advertiser.com. The third party sends its banner and set another cookie with a unique identifier, which is userABC. Both third parties now have set their own cookies. In the right part of the picture, you see the cookie synchronisation action. A user visits a third website, "website3", which includes an embedded gif image from the third-party tracker.com. The user's browser will make a request to this third party and send its cookie (identifier) with it. Now, the third party answers with an HTTP redirection URL. This response instructs the user's browser to make another request to the new URL. The user's browser sends a new request to this second third-party advertiser.com and includes the cookie identifier from the first third-party tracker.com and the cookie of the second third- arty. Advertiser.com now can synchronise both cookie identifiers to track the user over a greater portion of the web.

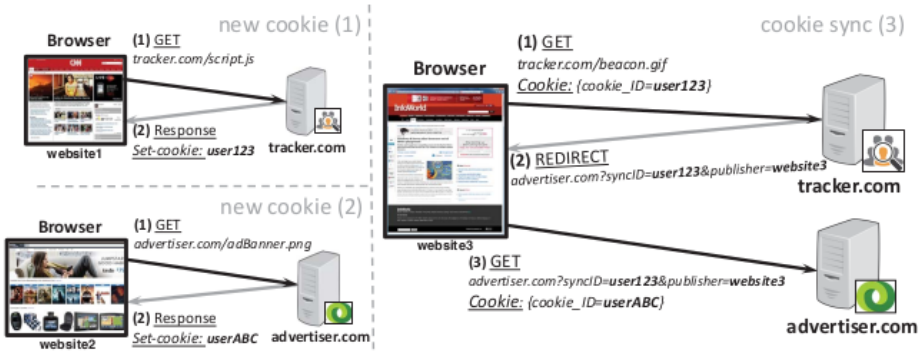

Figure 2.2: Cookie Synchronisation mechanism, Source: [PKM18]

INVISIBLE PIXELS

With the term invisible pixels, we commonly refer to 1 x 1-pixel images or images without any content [FBLS20]. They do not add any content to the web pages and allow third parties to use them for online marketing, tracking, and profiling the users to analyse their

behaviour [Dob10]. So, in general, the loading of this kind of images, which the third parties hosts, deliver tracking information to these third parties [RL18].

Invisible pixels are not new; websites used them for unobtrusive web tracking since the nineties [RL18]. They were, however, called differently. They started under the name web bugs. Dobias [Dob10] defined these web bugs as *"web-based digital tracking objects enabling third parties to monitor access to the content in which they are embedded"*. The mechanism remains practically the same up to today. The user's browser induces the HTTP requests automatically by, for example, opening a web page.

Two recent studies [RL18, FBLS20] have researched the use of invisible pixels for tracking in today's web. Ruohonen and Leppänen searched for invisible pixels by collecting the HTML code and extracting all the <a> tags (image tags). The src attribute within these tags contains the URL, and we can use it to determine if it involves a third party. They counted the 1 x 1-pixel images and concluded that 31 % of sites included at least one invisible image.

Fouad et al. used a different approached because they concluded that the method used by Ruohonen and Leppänen missed an important number of images that are dynamically loaded. Fouad et al. were able to get the total number of delivered images using the content-type HTTP header extracted from the stored HTTP responses. Their results show that 92.85% of all visited pages include at least one invisible pixel.

FINGERPRINTING

Every person on this planet has a unique fingerprint. We can use these to identify ourselves. We can apply the same principle to our computers and browsers when we surf the web. Although less unique, every computer has a unique IP address, and all browsers share a set of properties whose values differ from browser to browser. From the system and browser, a website can learn its properties that together form a unique or nearly unique identifier [MM12]. The results of a study by Eckersly [Eck10] show that 83.6 % of browsers were uniquely identifiable.

Torres et al. [TJM15] use the term fingerprint surface to refer to the set of characteristics that are used to fingerprint a user. We call these characteristics, more technically, attributes. The fingerprint surface can include properties or attributes, such as screen resolution, HTTP user agent and IP address, the HTML5 "canvas" element, the used Javascript engine, the fonts present, time zone, etc. Hupperich et al. [HTWH18] give a complete definition: *"Fingerprinting refers to the process of obtaining characteristic attributes of a system and determining attribute values that can be leveraged to recognise or identify a single system among others"*.

Although fingerprinting proved its benign use, such as presenting a mobile version of a site when the user browses with his mobile phone, trackers have successfully adopted the techniques. The fingerprinting trackers collect a visiting user's fingerprint and derive a unique identifier of it. This identifier can work just like an identifier from a cookie. Every time this user visits a website that embeds the same fingerprinting tracker, this tracker calculates the user's fingerprint and checks if it has already seen this user. If it does, then the tracker can construct a browsing history profile for this user.

Canvas Fingerprinting One specific type of fingerprinting is canvas fingerprinting. It uses the browser's Canvas API, which is used to draw graphics (images), on the fly, via JavaScript. We can further also use it to create animations or store content for gamers.

A fingerprinting script can use this Canvas API to draw invisible images on your canvas and afterwards, by reading these images back, extract a persistent and long-term fingerprint [AEE+14]. The hash calculated from these read-back images is the actual fingerprint. The fingerprint is unique for nearly every user because of the differences in font rendering, smoothing, anti-aliasing, and other device features, which causes devices to draw the image differently [EN16].

Acar et al. carried out a study to measure the extent to which websites use this type of fingerprinting. They found that 5.5 % of crawled sites run canvas fingerprinting scripts on their homepage. Later, and Narayanan modified Acar's measurement methods to reduce the chance of false positives. They found canvas fingerprinting on 1.6 % of the scraped websites, of which 98.2 % are from third-party scripts.

Figure 2.3 shows a graphical representation of the fingerprinting process. In step one, the script draws something on the canvas, for example, with the FillText() function. Next, it reads back the pixel data with the ToDataURL() function. Finally, in step three, the script calculates a hash of the data and stores it in the database.

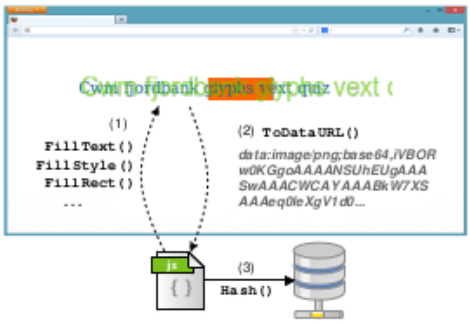

Figure 2.3: Canvas Fingerprinting mechanism, Source: [AEE+14]

2.3. Web scraping

Web scraping is the core technology that we use to perform our web measurements. Krotov et al. [KJS20] define web scraping as the practice of automatic extraction and organisation of data from the Web for the purpose of further analysis of this data. We can use web scraping to study a lot of different subjects on the internet. For example, the study by Blazquez et al. [BDGP19] used it to automatically monitor a firm's engagement in e-commerce by scanning their websites for specific keywords.

Web scraping offers many advantages, such as we can run the crawls at low infrastructure costs. Partly because of this, it is a core tool for online tracking research [ZBO+20].

There exist different types of tools, or web bots, that we can use to perform our web scrapes. The most rudimentary include Wget and cURL. Python offers us BeautifulSoup and Scrapy. More advanced options are PhantomJS and Selenium. Selenium automates a browser to load a page and allows us to perform actions on that pag [Mit18]. It allows to collect more accurate and complete data, but this comes at a computational overhead [ADZ+20]. We can use these tools as building blocks for other tools, such as OpenWPM, which became a web scraping framework based on Selenium and written in Python.

Although very popular, there is still a lot of controversy about the ethical aspects involved. On one side, it is certainly possible that a derived finding may unintentionally compromise the privacy of individuals [KJS20]. Furthermore, website owners see their website

data as critical assets, that they want to protect. On the other hand, these same websites (or some of them) engage themselves in web scraping. For example, Uber [Con70] has an extensive scraping program that tries to gather data about competitors, such as how many trips they were making. They, however, forbid the scraping of their websites to protect them from hackers and others doing the same type of information gathering.

WEB BOT DETECTION

A lot of website owners want to protect their assets, so they try to prevent web bots. Prevention starts with detecting whether a website visitor is a web bot. Over time, a few general categories of web bot detection techniques were developed. Two of those categories that are widely recognised are behavioural web bot detection and fingerprinting. Besides these, we all also encountered the CAPTCHA challenges. These image recognition tasks try to distinguish us between web bots and humans. Figure 2.4 shows an example of a CAPTCHA challenge.

Figure 2.4: CAPTCHA example, Source: [YA08]

Behavioural web bot detection exploits behavioural biometrics [CGW18] (observable human behaviour), such as mouse movements and keystrokes, to detect web bots. A server observes the behaviour displayed in the browser to detect deviations from behaviour that is difficult for a web bot to perform. The OpenWPM framework is based on Selenium, whose APIs (typing, scrolling, mouse movements) also exhibit very web bot-like and thus detectable features. Besides these browser behavioural features, a server can also record and analyse features that it extracts from a request, such as the number of pages visited.

We have already discussed the use of fingerprinting to track users. Web bots also have uniquely identifiable properties that we can fingerprint. The most apparent type of web bot that we can detect with fingerprinting is PhantomJS. Jonker et al. [JKV19], for example, researched the fingerprinting surface of web bots to find that fourteen types of web bots could be detected based on their distinguishable fingerprinting properties.

After web bot detection, a website wants to employ countermeasures to discourage the use of web bots. Such countermeasures can take specific forms. For example, Vlot [Vlo18] identified three types of categories. These are the blocking of the entire web page, block specific content, and display different content. Other research, such as that of Jueckstock et al. [JSS+21] confirms that the web behaves differently when we approach it from different measurement endpoints because of, among others, web bot detection.

USE OF WEB SCRAPING IN WEB PRIVACY STUDIES

Because web scraping is such an interesting tool to collect data online, it is used extensively in research studies. In particular, OpenWPM is an interesting web scraping framework for our research because it focuses on web privacy measurements, which connects closely to the context of our research. By now, about eighty studies have used the tool to aid in their data collection. For example, the study by Englehardt and Narayanan [EN16]

first presented the tool and used it to scrape one million websites to perform tracking related measurements.

They choose to use a cloud-based deployment to deploy their web bots. Other studies also choose cloud-based deployments, but other scraping starting points (or vantage points) are also possible. In fact, it is quite common that a study uses a combination of vantage points to study the different impacts on their measurements. Again, we see here a commonality between many studies that we will use in our research. Most studies perform web scrapes with multiple web bots to detect any influence of them on this variable to measure the impact of the main variable (of their research).

All web bots, scraping in the role of a client, make a request to a webserver that answers with a certain response. Different websites may use different webservers, but other external third parties, also hosted on a webserver, may also come into play. With web scraping, all requests travel through the internet, and this network also leaves its mark on the scraping results. For example, the location of the client and server plays an important role to target specific content, such as advertising.

In our research, these kinds of impacting factors are a central topic. The commonalities between the many scraping studies inspired us to design our model of a scraping setup that we will use extensively throughout our research.

3

RELATED WORK

A large body of knowledge already exists about user privacy, web bots, and their detection. We will discuss a selection of the most relevant studies that relate to our research. From these studies, we are mainly interested in the confounding factors and mitigations that they address. We will focus on the studies of the past decade preceding our research.

We try to cover a wider range of topics in this thesis. We are starting with an advanced tool, tailored to the specific requirements for privacy research, OpenWPM. Then we cover the most recent advances in the research area, using multiple web bots to compare the influence of different factors, such as web bot type. Finally, we look at some privacy studies concerning user tracking, price discrimination, invisible pixels, cloaking, and location-related studies.

OpenWPM: OpenWPM is our primary choice to implement our scraping model because of its focus on web privacy measurements and its time-tested generic tools. The first article that we will cite here is that from Englehardt and Narayanan [EN16]. They developed OpenWPM and used it to carry out stateless and stateful web tracking measurements on the Alexa Top one million websites. For this, they used an Amazon cloud service that employs a virtual machine with eight vCPU cores that could run twenty browsers in parallel.

Different web bots: Besides OpenWPM, there also exist other types of crawlers. The study by Ahmad et al. [ADZ+20] developed a web crawler comparison framework that they used to research the crawler capability and flexibility on how they impact the data gathering. The confounding factors that they deal with are [ADZ+20] concern the influence of location of the cloud infrastructure from where the scraping takes place—furthermore, the rapidly changing internet and how this manifests itself when your scrapes start at different times. And finally, IP address-based discrimination against IP addresses that are suspected to be associated with crawlers. They resolve these issues by systematically scraping with the eight web bots in parallel and dynamically assigning new IP addresses if their reputation declines.

Another study that compares the results of multiple scrapers or user agents is that by Pham et al. [PSF16]. They carried out a large scale study of website behaviour based on the responses to six different user agents. The confounding factors that they discussed included parallel scraping by launching six scrapes (one for each user agent) simultaneously.

They tried to hide their identity by launching the scrapes from different machines with different IP addresses. Above that, they spoofed the HTTP user-agent field in the HTTP headers for each probe. Finally, to avoid getting detected by any server that monitors the request frequency, they avoided visiting a site multiple times and allowed maximally five retries in case of errors (for example, connection errors).

Compare scrapes to less detectable version: We can compare the scrapes of different web bots. A few studies doing this found, that different web bots receive different website responses. The following three studies compare web bot crawls to human-like crawls.

The study by Zeber et al. [ZBO+20] considers the divergence of web crawlers to human browsing behaviour. The authors identified a list of issues that introduce variability. The paper explores issues, such as IP address reputation, differences between a cloud virtual machine, a residential, or a business IP address. They furthermore research the crawling technology, the underlying platform, the region, and statefulness or not. The team recorded approximately fifty two thousand real user's browsing behaviour for the human-to-crawl comparisons, using a specific tool, called WebExtension. A comparison of this data (or the aggregation of it) against the web scraping results retrieved by web bots revealed a significant difference in the number of visited tracking domains. Web bots seem to substantially visit a greater amount of this kind of domains.

The study by Jueckstock et al. [JSS+21] builds forth on the anecdotal evidence that the web behaves differently when approached from well-known measurement endpoints or with well-known measurement and automation frameworks. Their work presents the results of using their web measurement framework, which deploys six clients simultaneously. The clients vary in relative realism (human-likeness) of network vantage points (endpoints from which they visit the web pages) and browser configuration. The network endpoints are a university network, a residential ISP network, and a cloud provider's network. The authors mention concerns about the effects of geo-targeted web content. They further try to eliminate any sources of irrelevant differences by homogenising all controllable aspects of the crawls. For example, they performed all page visits so that they follow the same workflow and timeout limits. Another control involves doing three repetitions of the TRANCO Top 25000 websites' home pages (in a randomised order) to rule out any noise, such as the web's dynamism, content personalisation, and connectivity issues. A practical implementation includes using Cubernetes clusters for all network endpoints and synchronisation on the start of web page sets.

Another study that tries to compare web bot scrapes to scrapes of a less detectable web bot (is more human-like) is that of Krumnow et al. [KJK22]. This study investigates to what extent web bot detection influences web measurements or how websites react to OpenWPM after web bot detection. Examples of the confounding factors that they deal with are, because cross-client interference could block all the web bots run from a specific IP address, they used two different machines with different residential IP addresses. They also try to minimise the differences caused by web page dynamics, such as A/B testing, content updates, layout changes, and targeted advertising. They do this by scraping the same website eight times in parallel in a synchronised manner.

Invisible pixels: A recent study that picked up the wire on invisible pixels is that of Ruohonen and Leppänen [RL18]. The few things that this short paper mentions is that they

scrape a short list of websites, the Alexa Top five hundred. But, they also follow every link from the home page to the same domain. one confounding factor that we identified in their study is that they repeat their scrapes three times. They do this to rule out any temporary timeout failures.

The study by Fouad et al. [FBLS20] also studies invisible pixels. They first crawled the Alexa Top ten thousand (including ten links from each homepage) and analysed the requests and responses that lead to invisible pixels. Then, they defined a new classification of web tracking behaviour. This study's research method consists of scraping the websites twice (mainly to validate the cookies) using OpenWPM. To deal with the web's dynamic nature, they started the scrapes at the same time. Besides that, they performed the crawls on two different computers, using two different IP addresses.

Price discrimination: Price discrimination is another research area that uses web scraping as a tool to perform their measurements. Price discrimination refers to setting the price of a given product for each customer individually according to his evaluation for it [MGEL12], or based on the customer's personal information [HTWH18], or based on a customer's purchasing power and willingness to pay [VNBJ14].

One of the earliest studies on price discrimination is the study by Mikians et al. [MGEL12]. They tried to demonstrate the existence of signs of both price and search discrimination, facilitated by personal information, on the Internet. For this, they looked to three distinct vectors: technological differences, geographical location, and personal information. Their methodology consists of visiting two hundred vendors (three times), from multiple vantage points, for a period of twenty days. They used multiple local machines and six proxy servers to simultaneously visit a website to measure whether, amongst others, location affects prices.

Another study, by Hupperich et al. [HTWH18] that reassembles the previous one, studies price differentiation. The authors applied different fingerprints to simulate different systems and analysed any corresponding price changes. They applied their method to several booking websites and a rental car provider platform. They considered issuing queries from a different network location by using free proxy servers and rent VPN gateways to enable a flexible routing of requests.

The last study by Vissers et al. [VNBJ14] searched for price discrimination in airline tickets. The authors analysed twenty-five airlines for three weeks, using sixty-six unique user profiles. These user profiles emulated real users and were constructed in the light of user tracking. They used a CasperJS scraper (based on PhantomJS in headless mode) to query the airlines from two locations simultaneously. They also spoofed their user-agent and navigator object to measure any price discrimination based on their Operating System (OS). A few actions also reveal that they took care of some confounding factors. For example, they followed exactly the same steps as a normal user would do when searching for airline tickets. They randomised the order in which the user profiles were used for each scrape to prevent any possible detection of the used sequence of the profiles. And finally, they did not run their scrapes in parallel to not disrupt the airline server and prevent detection in case the server employs any form of rate-limiting.

Cookies: Acar et al. [AEE+14] present the first large-scale studies of three advanced web tracking mechanisms. These are canvas fingerprinting, evercookies and the use of cookie

synchronisation in conjunction with evercookies. In the experiment on Canvas Fingerprinting, they scraped the Alexa Top 100000, using up to thirty browsers in parallel. This number helped to reduce the crawl time. Besides this, they gave much attention to eliminating false positives to rule out benign scripts that use the Canvas API. In the study on cookie syncing, the authors used Amazon EC2 instances to scrape the Alexa Top 3000. They performed three crawls, each with different privacy settings (e.g. allow or block cookies). Although they perform multiple scrapes, some even in parallel, they do not mention anything about any synchronisation between them.

Mayer and his co-author Mitchell [MM12] present a survey on the policy debate surrounding third-party web tracking. They based much of their work on the results of a new web measurement platform that they developed. One specific feature of the platform is that it uses a production web browser, which closely emulates real-world browsing.

Cloaking: Cloaking refers to the practice that web servers send different web content to web browser than to search engines. With this search engine, we mean a kind of crawler (or spider) that collects data about a web page.

One of the first studies that tried to identify cloaking, is that of Wu and Davison[WD05]. To test for cloaking, the researchers crawled a selection of web pages simultaneously from two IP addresses with a spoofed user-agent HTTP header. One university-based and one commercial IP address. They performed this crawl two times within the time interval of the same day. The comparison technique consists of comparing two web bot crawls to the results of a human visiting the site with a browser.

A similar more recent study by Invernizzi et al. [ITK+16] developed a scalable de-cloaking crawler and classifier that detects when a web server returns divergent content to two or more distinct browsing clients. Their setup consisted of eleven distinct browser and network configurations to perform three stateless crawls each, in order to trigger cloaking logic. They repeated each crawl three times to rule out any noise introduced by dynamic content or network errors. The distinct configurations included the use of three vantage points (a search engine, a browser, and a stealthy deployment using mobile and residential networks), and furthermore three native platforms (Chrome on Desktop, Chrome on Android, and a basic HTTP fetch). The IP address setup, aimed at keeping a clean reputation towards the cloakers, consists of proxying network requests through a tap that provides a configurable exit IP. This IP belongs to either Google's network, a mobile gateway, or a residential IP addresses.

Location based privacy regulatory model: Fruchter et al. [FMSB15] tried to measure the impact of privacy regulations in different countries on the amount of tracking and advertising in these countries. To do this, they deployed four AWS cloud-based virtual machines, running OpenWPM. The use of OpenWPM allowed for synchronisation between the scrapes. The authors deployed their setup from four different AWS servers, in four different countries. This approach makes it unnecessary to use VPN proxies from these countries to mimic local requests. With this setup, they scrapes a small selection of websites, the local Alexa Top 250 from each country from where they scraped.

4

Research Methodology

Our research involved carrying out many different tasks. We present our methodology to answer each research question.

First, we discuss the motivations behind our research approach. Afterwards, we present the model of a scraping study that we will use to structure our thinking further. After that, we discuss the methods that we used to answer the research questions. The first two research questions concern the impacting factors and mitigations. Because we largely followed the same research method to answer them, we discuss their methods in one section. The following section will discuss the methods we used to develop the taxonomy. The final research question, concerning quantifying the impact of the factors and mitigations, is also based on a literature study. We briefly discuss how we evaluated the quantification methods.

Motivation for the research approach

There exists an optimal approach to answer each of our four research questions. We will briefly discuss the logic and our motivations to use our approach by comparing our it to some alternatives.

We, first of all, have our impacting factors and mitigations We chose to perform a literature study because it was the most productive method that allowed us to perform a thorough job and we get a form of validation because of the acknowledgement of the factors in the scientific literature. Furthermore, depending on the literature studies that you use, you can widen the scope of your impacting factors. Alternatively, we could try to design a scraping setup for a scraping study ourselves and record what problems we may have encountered. But how many problems would we be able to identify by ourselves? It would be very unclear what the outcome would be and, besides of that, there is little guarantee on success. Another downside with this approach is that the scope is very small and it would take to much effort. We choose the same approach to answer the research question on the mitigations because the same reasoning applies to the mitigations of the impacting factors.

Following the discovery of impacting factors and mitigations is the categorisation of them. We quickly understand that not all factors are equal. Although we find similarities, there also exist big differences between them. For example, we know two forms of web bot detection: browser fingerprinting and behavioural web bot detection. But there also exist very non-similar impacting factors that can impact our web scraping results, such as

client-side errors and changes in the web page content (or content updates). With the categorisation, we want to bring some order in the set of impacting factors so we can quickly get an understanding of which factors resemble which others. There are several approaches to create a categorisation, each one more suitable for a specific situation than the others. We could, for example, use the co-occurrence technique that groups impacting factors together that co-occur in a given text. However, we also noted large differences between the impacting factors that research studies mention together. We could furthermore choose to make a classification, but for this, we would expect that we identified every impacting factor that exists. And we do not have the pretentiousness to say that we were able to do that. To get an overview of the complexity of a scraping study, we started with a domain analysis that resulted in a generic scraping model. The model allows us to identify additional impacting factors. It further also can help us to find categories for those impacting factors. The scraping model provides a good basis for developing a taxonomy, which is our approach to creating the categorisation. We can easily incorporate the categories that we derive from the scraping model into our taxonomy. Other advantages are, that as it can help us to identify impacting factors of which we would normally not think about.

For the final research question, we wanted to analyse quantification methods to measure the impact of our impacting factors and mitigations. We see two general approaches to answer the research question. One is to start by writing down the requirements for a quantification method. Then, we could consult the statistical literature to find any techniques that we can use in a complete quantification method. Unfortunately, it would be very hard to find out the requirements from past research studies. We also question what the final suitability of such a method would be. Our approach consists of looking at what others used to quantify the impact of their main research variable. This variable is not infrequently also an impacting factor in our research. We scrutinised their methods and searched for ways to improve it. The advantage of our method is that we get insight into the real research's common practices, which we can later process in our proposal. This will ensure that our method is not totally irrelevant.

GUIDING SCRAPING MODEL

There are many scraping scenarios possible to construct a scraping model. For example, scraping from the cloud or by a group of researchers from a university network. We want to keep up with the most recent advances and challenges in web scraping studies. Recent studies, such as the of Zeber et al. [ZBO+20] and Jueckstock et al. [JSS+21], based their research on anecdotal evidence that websites respond differently to web bots than to human visitors. Their measurements, for example, confirmed that the amount of visited third-party trackers differs between the two types of visitors.

Figure 4.1 depicts, very plainly, this problem. A human visitor and a web bot, which we can also design in such a way that it acts very human-like, make the same kind of request to the same website. The website, however, makes a calculated distinction between the kinds of visitors. Based on this distinction, the website returns different responses. This issue does not stop with this one distinction. Websites seem to be able to distinguish between different kinds of web bots too. [ADZ+20]. So different web bots would receive different website responses. For example, a Wget-based web bot gets different types of responses than a PhantomJS-based web bot.

We assume that if a study employs a web bot, then the issues that we just mentioned

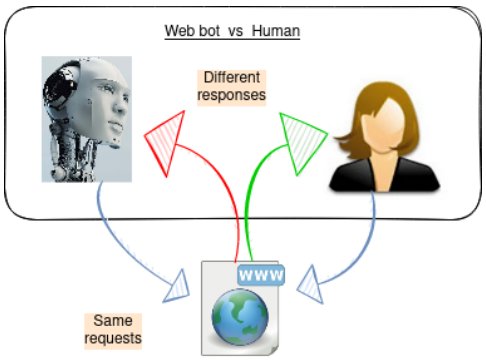

Figure 4.1: Websites return different responses to web bots than to humans

might also affect that study (if the differences are relevant). Furthermore, if a study employs multiple web bots, then each could get different results. But there also exist other reasons that cause a web bot to get different website responses. For example, if we deploy different web bots from different countries, then they could get different website responses due to geo-targeting. We refer to such factors as impacting factors, and our research tried to search as many such impacting factors as we could.

In our research, we want to base our findings on real-world scenarios. For this, we designed a model of a scraping approach that will guide our reasoning and discovery of the research results. We based ourselves on the state-of-the-art research studies that utilise web scraping tools. We took the commonalities from their web scraping approaches to keep ourselves up to date with the challenges involved in these scraping studies. We will specifically use these commonalities to limit the scope, identify relevant impacting factors, and derive categories for our taxonomy.

Figure 4.2 shows our model of the scraping approach. We kept the model generic so that it allows for some flexibility in its implementation. This way, it can apply to different web measurement studies.

In its most basic form, a web scrape consists of a client who requests a web page from a server. This request and response pass through a network, the internet. In this model, we recognise these three core concepts: client, server, and network. An additional fourth major concept consists of all those impacting factors that represent the major choices that we can make to design our scraping study. Such design choices may also impact the website responses. The scraping study design part of our model functions as the extensions of the web bots in the client-side part. Although very closely related to the client side, the choices are made independently enough to become a separate concept.

The client consists of multiple web bots, among which, a less detectable web bot. With the less-detectable web bot, we aim to imitate a human visitor. We would prefer to compare website responses to web bots against website responses to human visitors. However, we cannot instruct humans to visit thousands of websites, all simultaneously. Besides that, humans do not scale very well. We use multiple web bots so we can compare the website responses to each web bot. If we get different website responses, it might possibly be due to one of our identified impacting factors. In the model, the different web bots make their requests to the website. The website is located at the server side of the model. It may return different website responses that may result from, for example, the dynamic web. We represent this with the coloured arrows returning from the webserver through the network

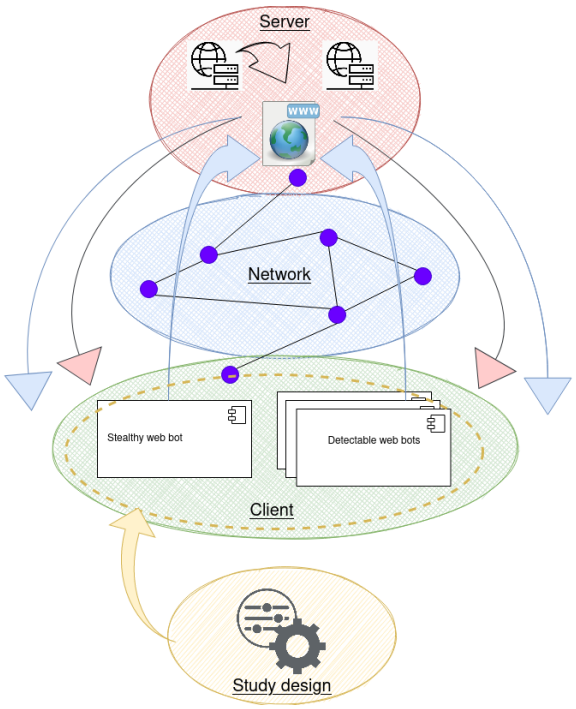

Figure 4.2: Model of scraping studies

to the clients (web bots). The network, in between the client and the server, with all its related technologies, can also influence the website response. For example, a website can derive a lot of information from the client's IP address, such as the approximate location.

IMPACTING FACTORS AND THEIR MITIGATIONS

In this section, we present our methodology to find factors that impact the results of a scraping study. Those factors would cause some web bots to receive different website responses. Differences in website responses might also impact the scraping study's measurements. We want to limit our scope a little more because, these days, many researchers in different fields found web scraping a handy tool for data collection. For that, we selected literature about scraping studies, which perform web privacy measurements, and preferably compare web scrapes. We have already introduced the selected literature in the Related work section. We used these resources to perform a structured literature review of the confounding variables identified in these studies.

In addition, preliminary brainstorming sessions did already result in a list of confounding variables. This brainstorming involved the scraping model that we just discussed. We consider the people involved in this brainstorming exercise to be subject matter experts. Because of our brainstorming, which yielded a substantial amount of impacting factors, we could think that a short literature review would not add much value. However, we found some benefits using this approach. First of all, we do a more thorough job, looking at studies that measure different things or put different accents on the object that they study. Second of all, we learn which factors a realistic study implements.

Our methodology for the literature review is to, first of all, get a broad overview of the article's purpose. Next, we look straight for the methodology. We read through the methodology and tried to identify any confounding variables. Most studies also discuss a few con-

founding variables in a separate subsection. From such sections, it is easy to identify them. So we identify and note the confounding variables. The discussion sections could also be the source of such confounding variables. However, we believe that we would need to search them between the lines, which would yield very subjective results, so we did not use such sections, unless they were explicitly mentioned.

The studies' discussions of the confounding variables are often accompanied by the researchers' approach to mitigating their influences. We note each study's approach for all of their confounding variables and use this as the basis for a discussion of research question two. Besides these tasks, we searched for extra resources on the internet for each impacting variable that we identified. We do this to determine the variables' importance and dismiss the trivial variables.

We also asked ourselves how we could handle unknown impacting factors? Should we not have to know the impacting factors before we can handle them? Based on the theories of taxonomies, we described a method to search for new impacting factors. Furthermore, after experimenting with a few ideas, we used the resulting knowledge to write down some steps to handle any newly found and still unknown impacting factors.

Creation of the taxonomy

After we identified a list of impacting factors, we would like to present them. Khan [Kha17] defines taxonomies as systems that we can use to classify and organise things. A taxonomy will make it easier, also for other researchers, to afterwards, find additional impacting factors. We say that we classify things, but a taxonomy is something very different from a classification system. According to Hedden [Hed20], a classification system differs from a taxonomy in, among other things, that it is comprehensive and it covers everything in its sub-domain. While a taxonomy, on the other hand, has its terms hierarchically related to each other. We do not consider a taxonomy as a finished object. It is a living document that we, in collaboration with other researchers, can extend by discussing its content and making improvements based on the experience of using the taxonomy.

We dropped the term hierarchy, which is one of the types of taxonomies. Other types are flat, network, faceted, but the hierarchical type fits the best in our case, because this type of taxonomies is suitable for smaller taxonomies. There are often different ways to create a hierarchical taxonomy. One more complex type of hierarchical taxonomy is called a polyhierarchy. This type allows a term to have multiple broader terms [Hed10].

There are basically two working methods to create our taxonomy. These methods are top-down and bottom-up. We already have identified our list of terms (impacting factors) that we want to answer the first research question. The most straightforward approach would be to use the bottom-up method and group our terms into categories. However, different sources [Kha17, Hed20] advise combining both approaches and integrate the results in the middle of the hierarchies. For the top-down approach, we will base ourselves on the scraping model, from which we will derive the top-level terms of our hierarchy. These approaches also correspond to the viewpoints of lumpers and splitters. We split our high-level terms into less general (or narrower) terms, and we lump our lower-level terms together into categories.

Our hierarchical taxonomy consists of terms. Higher-level terms are the broader categories, and the lower-level terms are our identified impacting factors. We cannot include everything we can think of in our taxonomy. We have to be selective because if we use too

specific terms, then they will not match with what the user would look for. The consequence will be that users will not use them.

We choose to use the selection criteria for terms presented in the book "The accidental taxonomist [Hed10]". We repeat the four main points. The first criterion is that they must lie within the subject-area scope of the taxonomy. Secondly, the concept must be important enough (it is mentioned in multiple sources) so that users will likely look it up. Thirdly, there is enough information on the concept. And finally, users want and expect that the concept is covered.

To finish our taxonomy in this thesis in the most complete state, we extensively reviewed it with the subject matter experts.

QUANTIFICATION OF THE IMPACT

The literature review brought forth a set of impacting factors and mitigations. Some studies that study these impacting factors as the main variable also discuss their research results. They use more advanced statistical quantification techniques to show their results and draw appropriate conclusions. We base our answer to the last research question on these studies.

We will work as follows. First, we shortly present the quantification method for a handful of studies, then we will evaluate them based on a few simple criteria. For the first part, we already clarified the context of the study and their scraping set up in the related work chapter. We will just shortly mention the type of measurements that the study performs, for as far as these are relevant. Then we will briefly describe how they quantify and present their results, to which we add a summary of the major findings.

We then proceed to evaluate the quantification technique by holding them in the light of three points of interest. These points are, what are the pros and cons of the methods. For example, can we easily understand the statistical methods that the studies use without a statistical background, and can we quickly draw conclusions from the given results? We finally evaluate if we can readily apply the same techniques to the other identified impacting factors and mitigations, or in other words, are the methods generic?

We finish with a recommendation for a quantification method that is generally applicable to the identified impacting factors. This method should keep all the positive aspects of the studied methods while avoiding the negative aspects. The purpose of the method is to quantify any differences in measurements due to the impact of the impacting factors and mitigations. But we must not only compare data, we must also be able to draw conclusions from them at a high and a lower level.

5

IDENTIFIED IMPACTING FACTORS

Most scientific studies recognise the existence of confounding factors. These confounding factors can impact the specific study that mentions them, but they can mostly also impact a scraping study in general. They thus can affect the scraping results.

We can define what this kind of phenomenon that causes an impact on a study's results (or, in other words, it affects its results) is about. An impacting factor generally implies that one kind of factor is the cause that a web bot (or even a human user) receives a different website response than another web bot. We base this definition on the scraping setup that we implemented in our scraping model. However, we kept our scraping model generic so that it would apply to other scraping studies as well. Besides that, we encountered impacting factors in different scraping studies that do not really fall under this definition. For example, we know that websites change their content over time, so incidentally, two web bots could scrape different content (one the old content, the other the new content). But it is also very well possible that a single web bot scrapes new content on subsequent web scrapes. Above that, differences in website responses due to web bot detection could very well be possible because one web bot visits a website too many times in a short period.

So we need to loosen the definition a bit, and we want to add one more variation that allows other researchers, with other study designs, to add impacting factors to our taxonomy. For impacting factors in the study design category, we accept them as impacting factors if a design choice affects the website responses. So, different design choices imply different website responses. The definition then becomes: an impacting factor is a factor that causes deviations in website responses for different web scrapes, whether they are performed by one or multiple web bots (with one or another setup configuration).

This section lists those impacting factors we identified, accompanied by a short explanation and the reason why we include it in our collection. We group our impacting factors according to the categories that we identified for our taxonomy. We will discuss our taxonomy in the next chapter. The categorisation, here, can structure our reading a bit and make it less dull than a plain list. Our four main categories are the factors related to the scraping study design, and furthermore, everything related to the client side, the server side, and anything related to networking in general.

5.1. FACTORS RELATED TO THE SCRAPING STUDY'S DESIGN

CONFIGURATION PARAMETERS

Bias in selection of websites: Website selection in privacy studies is largely guided by the use of lists of top rating websites. Through the literature study, we noticed a general preference for the Alexa top ranking websites. Recent studies, on the other hand, prefer the TRANCO list because it is less vulnerable to rank manipulation. As such, it is a more reliable list. The used range of top-ranking websites differs substantially from two hundred fifty [FMSB15] to one million [EN16]. Besides that, some studies, such as that of Krumnow et al. [KJK22], make a selection of a larger list. Top ranking websites receive more user traffic than lower-ranking websites. More traffic leads to greater profit potential for advertising and gives a greater incentive to track users. We conclude that limiting a selection of top-ranking sites may introduce a bias in privacy measurement results. We consider this an impacting factor because different scraping runs with different selections of websites (scraping study configurations) will result in differences in website responses.

Normalised URL format: A URL can get complex because it can be composed of different parts, such as the protocol, a hostname and a file name. We see a long version in the demo tryout of OpenWPM (demo.py), which includes the protocol and the scheme (HTTPS, www). In the short version of the TRANCO list, the URL only consists of the hostname. If we would like to rewrite these short URLs to include the protocol part (HTTP/HTTPS), we would obviously use an automated method. In this case, we would rewrite all URLs to include the same protocol. Most websites use the secure version, but there are still websites that keep using the insecure version. From the short URL version, we cannot know what the correct protocol is, so we would be forced to use the same version for all websites. Previous experiments showed that some websites fail to load if you use the wrong protocol. Again, different scraping configurations can use a different (long or a short) URL format, which will be the cause of different scraping results.

Amount of scraping traffic: One step that we can take to reduce our footprint towards web bot detection mechanisms is to use a less detectable web bot. Vissers et al. [VNBJ14], for example, also take web bot detection into account. Their scrapers follow precisely the same steps as a regular user would follow when manually searching for airline tickets. With this, we raise the bar to get detected as a web bot. If we raise the bar, we should not ruin it later by performing very suspicious things, such as performing many parallel scrapes from the same web bot. Such things seem suspicious because a webserver will track the number of requests you make in a certain period of time. The amount of scraping traffic is a subject of discussion in many studies, such as that of Krumnow et al. [KJK22], who perform eight parallel scrapes. Other studies, such as that of Pham et al [PSF16], are much more careful and perform a second scraping run, only after a certain amount of time. This is an impacting factor in the sense of the variation of our definition, where subsequent web scrapes can cause website response deviations (only after too many web scrapes).

Equivalence of web scrapes: We base our research on a model of a scraping study that deploys multiple web bots. The design of the scraping study must take into account that the website serves every web bot a different version. If all web bots perform the web scrapes

in the same manner, then it is easier to understand what you are scraping. Furthermore, numerous studies do not only visit the website's homepage; they also select links to visit more web pages and so to perform a deeper web crawl. Englehardt and Narayanan [EN16] argue that, by performing deeper crawls, the average number of third parties increases. In the case that we use multiple web bots, we must take a closer look at our website links selection strategy to make sure that we select the same links for each scrape. If we select different links each time, then this might make it more difficult to compare the results from the different crawls. Difference between web bots can be the subject of an impacting factor, but in this case, it is the difference between web scraping configurations (study design). Different configurations lead to different results.

5.2. Impacting factors related to the client side

Web page dynamics

Cookie Dialogues: Cookie dialogues are those popup screens asking you to accept that the website places cookies on your computer. We mostly encounter them when we visit a website for the first time. In general, they allow you to choose between two options, to accept them all or more advanced settings. When we select the advanced settings option, you can choose the types you allow, such as the essential cookies, analytics, personalisation etc. Cookie dialogues have different appearances in the HTML code, such as simple div tags and iFrames. Dialogues can block the entire page and disable (or blur) any content depending on the website. This impacting factor could also fit in the scraping study design category because we expect a different configuration to cause differences in web scraping results.

Live updates: Many websites want to sent their clients the latest updates. They can use the newest notification techniques for this. Twitter, for example, sends you the latest tweets from the people that you follow in real time. You do not even need to update your browser window. We call such techniques live updates. After you load such web pages, when you scrape with your web bots, it can load additional content. This issue is an impacting factor because one web bot on subsequent scrapes or even two web bots (almost) simultaneously can scrape different content because of the live updated content.

Infinite scroll: Facebook and Reddit are just two of many websites that initially only load a part of its content. On such sites, users share their content, such as pictures and messages, with others. These messages are ordered chronologically, such that, when the page first loads, you see only a handful of the latest posts. Subsequently, when you scroll down, you load more (older) content. Since, over time, there exist a long accumulated history of posts, it seems as if you can scroll infinitely to load extra posts. If you scrape such sites with your web bot and want all the website's page content, you will have to let your web bot scroll down. A single scroll down the page will not do the job, because the extra content only loads incrementally. A solution to scroll infinitely on some websites but not on others is not very scalable. So we have to choose one configuration for all the websites together. This is thus an impacting factor because differences in scraping configurations (study design) would lead to different website responses.

Errors

Browser errors and crash: We most probably use a browser-based web bot to perform our web scrapes. One example is OpenWPM, which uses a fully-fledged Firefox browser to scrape the web. This approach has big advantages, such as appearing like a real user towards a website. Although widely tested and stable, there might occur some problems after long periods of operation [EN16]. This factor relates to any errors occurring in the browser that we use to scrape the web. If one browser crashes, then this web bot's scraping results will be much different than those of a later, successful web scrape by the same web bot or by other web bots. Because of these differences, we consider all kinds of errors, impacting factors.

Scraping framework crash: We can perform web scrapes with simple scripts, but we prefer to use a scraping framework, such as OpenWPM, to make more accurate and complete measurements. OpenWPM can start any browsers that may crash. Worse would be that the whole framework crashes. We generally do not expect such thing to happen, but errors and crashes are just an unpredictable part of life. This error is an impacting factor for the same reason as the previous, although it is much worse.

Scraping database crash: Just as it is possible that the whole scraping framework crashes, it is possible that any external services that it uses, crash. In this case, we could be talking about the database that OpenWPM uses to store its scraping results. This error will have consequences for any web bot that uses the database. Again, possible differences with web bots that are not connected to this database make us consider this an impacting factor.

Whole computer system slowdown or crash: Computer crashed got quite rare these days (even for Windows operating systems). Slow systems, on the other hand, irritate us frequently. It will make you wait for whatever you are doing. Slow systems could influence the loading of the web page, which might influence the completeness of the collected data. Delays may cause one web bot to scrape less content, which differs from the full content, so this is an impacting factor, just like the other factors in this category.

Attacks

Attacks on OpenWPM data collection mechanism: Because OpenWPM is probably not widely known by the general public, it has not gotten much attention from hackers. So we do not have to worry about any widespread exploitation of its vulnerabilities. Since it is predominantly the privacy and security research community that uses the tool, there is plenty of expertise to search for vulnerabilities. The research study by Krumnow et al [KJK22] dedicates a part to the discussion of a few vulnerabilities. For example, because the DOM event dispatcher is not protected, blocking the data collection mechanism and injecting false data is feasible. It seems further also possible to attack the completeness of the measurements. This kind of attack can target OpenWPM, but other types of web bots may be spared, so there can be differences if different web bots scrape the same malicious website. We can call this an impacting factor because of these possible differences.

WEB BOT DETECTION

Browser Fingerprinting: The background chapter did already explain what browser fingerprinting is. Websites try to learn your unique attributes, create a fingerprint, and then try to re-identify you on subsequent visits. Fingerprinting allows for tracking users but, it is also one of the mechanisms to detect web bots. Jonker et al. [JKV19] have studied the fingerprints of various web bots. They found that there exist a set of fingerprints that can specifically identify a web bot. Different types of web bots exist, such as simple web bots, that are easily detectable and more advanced web bots that use a full-fledged browser. These advanced web bots, such as OpenWPM, are also detectable. Krumnow et al. [KJK22] have researched the detectability of OpenWPM's attributes in their study. They show that specific attributes, such as using Selenium, is widely searched for by web bot detectors. In addition, they found that specific OpenWPM versions, such as the headless, Xvfb, and the virtual web bot, have additional fingerprinting attributes that make them even more detectable. We see an impacting factor in this because different web bots will have fingerprints that are more or maybe less detectable. A less detectable web bot will be less prone to web bot detection and its accompanied countermeasures. So differences between web bots can lead to differences in web scraping results.

Behavioural web bot detection: Next to fingerprinting for unique attributes, a web server can also collect information about a user's behaviour on the website. The primary user events of interest to the web server are keystrokes, mouse movements, and scrolling. Different studies have proposed web bot detection mechanisms based on behaviour. For example, Chu et al. [CGW18] developed a detection system based on machine learning classification. They could differentiate between human behaviour and web bot actions because web bots perform actions in regular patterns of limited variability. While on the other hand, human behaviour is characterised by irregularity and burstiness. For example, the cursors of web bots always move in straight lines and at constant speeds. Krumnow et al. [KJK22] additionally indicate that typing speed of web bots is much faster than that of humans, and that scrolling typically consists of one event. We mainly see an impacting factor in this because of the differences between humans and web bots, unless different web bots perform different kinds of behaviour on the web pages. But this then violates our assumption that we should only compare equivalent web scrapes.

HISTORY

Scraping history: When we manually visit web pages, we accumulate data in the background, from which the most notably, cookies. Websites also set and retrieve cookies when we scrape them with our web bots. Websites and third parties use the cookies to, among others, construct a user profile based on the history of past website visits. For whichever set of websites that we scrape, we accumulate a history and a particular profile. Or maybe not, because OpenWPM allows us to specify a configuration to delete all previous history after each scraped website. Normally we would not want any interference of the influence of history on our measurements. On the other hand, for some studies, such as cookie synchronisation, we specifically need this accumulation of history because we can then witness the sharing of the identifier cookies. Furthermore, some studies that measure price discrimination, such as that of Vissers et al. [VNBJ14], try to accumulate a user profile to provoke different prices based on that profile. One web bot that repeats a stateful web scrape (one

that accumulates history) to the same website may scrape different results than on the first visit. So differences in subsequent scrapes make this an impacting factor.

Client-side caching: Client-side caching refers to a technique that stores parts of web pages in a persistent memory on the client's computer. We do this so that, on subsequent requests, we do not need to request and load all the same data from the server. The most prominent advantage of such a technique is that it optimises the load time of web pages. It further then also increases its responsiveness. The data that the users mostly store are images, videos, and HTML documents. Other uses, according to the Mozilla developer web docs [1] are, that it allows us to persist previous site activity and we can save documents, generated by a web application locally for use offline. This is an impacting factor for the situation where a web bot (or multiple) performs multiple web scrapes. So subsequent scrapes will be affected by the cache. Or we could also compare the use of a cache against a scraping run that does not deploy the cache.

5.3. Networking-related impacting factors

Errors

Network errors: The World Wide Web consists of billions of web pages targeting a plethora of users. Most of them, far away on other sub-nets. Likewise, in our area, there can be much traffic on the internet infrastructure. We further on have our own Local Area Network (LAN) at a smaller scale with its peculiarities. On the scale of all these networks, with so much complexity, errors occur frequently. There exist many different types of errors related to the involved networks. Every protocol on the whole internet protocol stack has its own set of errors. We will mainly base our discussion on the errors produced by the HTTP protocol. Networking errors can impact the web scraping results of one web bot, but maybe not on those of another web bot or on subsequent scrapes. Because of these kinds of differences, we consider this issue an impacting factor.

Request/ response modification

ISP advertising: The Internet Service Provider (ISP) will manage your access to the internet, assign you an IP address, and forward your traffic over the network. Around 2007, some IPSs got interested in the revenues that online advertising companies gained from behavioural advertising. Behavioural advertising is about collecting information about a user's online activities to later send them targeted online advertisements [CDT08]. Since an ISP forwards all the user's internet traffic, they have access to the entire stream of the user's web use. To leverage this information, ISPs started to collaborate with companies, such as Phorm, NebuAd, and FrontPorch, that all use deep packet inspection techniques [MWRK10]. The use of software, such as NebuAd, allows an ISP to inject its own targeted advertising into the web pages instead of the normal ads. For example, an ISP Redmoon, used this software to hijack pages, to place its own ads on the page [2]. Such techniques have obvious privacy implications, and in the USA, they violate wiretap laws. Maybe in your scraping setup, you deploy one web bot that uses one ISP and another web bot that uses another ISP

[1] https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Client-side_storage

[2] https://techcrunch.com/2007/06/23/real-evil-isp-inserted-advertising/

that injects ads into the packets that it forwards. Because of differences in web scraping results that this may cause, we also call this an impacting factor.

Network caching: The functionality of a network cache resembles a lot to that of a client-side cache. Network caching solve the new network challenges associated with the increase in internet traffic. Now the caches are located near the client and they act like an intermediary between the browser and origin server. Again, the cache stores web content and it serves this content upon later requests. Advantages of this type of caching are [3] accelerated content delivery and it minimises redundant network traffic. We can think of this as an impacting factor for the same reasons as we consider client-side caching an impacting factor.

Middleboxes: We find middleboxes mostly in enterprise networks and ISP, where they perform in-network functionality. Carpenter and Brim [CB02] define a middlebox as *"any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host"*. Other operations includes among others, functioning as a firewall, load balancer, and intrusion detection system. If you plan to perform your scrapes from within an enterprise network, your scraping traffic might pass through such a middlebox. When we perform web scrapes with multiple web bots, of which one in such a network with middleboxes, then we might isolate differences in web scraping results that these middleboxes caused. That is why we call middleboxes an impacting factor.

CLIENT-LOCATION

Vantage point: With the term vantage point, we aim to indicate from what kind of a network the scraping requests originate. What kind of IP address do we own within this network? There are many examples of different types of networks, and we do not intend to give a comprehensive categorisation here. Examples are residential networks with residential IP addresses, corporate (businesses) networks, university networks, proxies, etc. For our discussion, we are mainly interested in the distinction between residential IP addresses and the rest. Research, such as the study by Jueckstock et al. [JSS+21] have already shown that web bots that scrape from residential IP addresses receive different web scraping results than web bots that scrape from cloud-based IP addresses. This then is an impacting factor because different web bots that scrape from different locations receive different web scraping results.

IP address blacklisting or blocking: For every HTTP request you send out through the internet, you send your IP address along. The web server needs this piece of information to know where to send its response. The HTTP header contains this information. Some web servers or services work with special lists of IP addresses that they tagged for some reason. These reasons can be, for example, suspicious behaviour in the past or an IP address of a well-known scraping server. Your IP address could also end up in such lists and get shared with others when you appear suspicious to a webserver. IP blacklisting goes hand in hand

[3] https://www.cisco.com/c/dam/global/de_at/assets/docs/Net_Caching.pdf

with other countermeasures, such as block pages. If we have one web bot that gets blacklisted and confronted with possible countermeasures and another web bot that performs successful web scrapes, then we have the same kind of situation as with errors. We can thus call this an impacting factor for the same reasons.

DNS resolution: DNS stands for Domain Name System. It is the protocol that translates the human-readable domain names to IP addresses. The IP address is written in a specific format that is easily readable by machines and allows to determine the client's location. DNS resolution refers to this translation process. To resolve a hostname, a client will query a DNS server that stores the DNS records. The DNS server actually consists of a hierarchy of distributed servers. Usually, your ISP will assign you a DNS server in your proximity (automatically through the DHCP protocol). The communications with the DNS server takes place through a DNS resolver, which acts as the client side in the resolution process. Communication with the DNS resolver takes place through the web browser (or web bot). Web bots located in different geographical areas will query different DNS servers that might redirect the web bots to different local versions of websites. In this case, this is an impacting factor because different web bots in different locations scrape different results.

Geo-targeting: We, who are living in the European countries of the west, share the same democratic values. There also exist more authoritarian countries, such as China and Iran, which, in their turn, value other norms above freedom of speech. They take more control of the internet to block any foreign influences. On the other side, we also do not want any jihadi websites on our web servers. So back and forth, countries can decide to serve and allow different content to and from specific other countries. Much less dramatic is the use of location to target advertising. For example, if we use a VPN proxy through a Swiss server, we might get ads about Swiss yoghurt, while if we used a British proxy, we could get ads about British dog food. Furthermore, studies, such as that of Hupperich et al. [HTWH18, MGEL12], show that commercial websites also use location to serve different prices to users from different countries. Geo-targeting also affects user tracking. Fruchter et al. [FMSB15] for example, discovered that geo-targeting can target a client's location, but the privacy regulations from the servers' location do certainly also affect the website's advertising and user tracking practices. Here, we see that a scraping setup with different web bots that operate from different locations, receive different website responses. Such causes of differences is what we defined as an impacting factor.

SERVER-LOCATION

Content Delivery Network/ Datacentre: Arie [Ari21] describes a Content Delivery Networks (CDN) in her blog as a globally distributed network of web servers (or Points Of Presence (POP)). These servers distribute the web content from locations close to the client. Making a request to content providers on the other side of the world can take some time. For larger web pages, you may need multiple requests, for which latencies can add up to some serious delays. A CDN's primary purpose is to reduce the latency by hosting the content in distributed caches worldwide. This way, there is always a server close to the requesting client. From these points, there is much less latency to request your web page.

Besides content caching, a CDN can provide additional services. For example, Fastly [4], a CDN service, provides authentication and other segmentation services. These segmentation services can consist of, among others, A/B testing and tracking related services. Not all datacentres are CDNs. In our research, we will refer to a privately owned CDN as a datacentre. Companies such as Amazon, Google, and Facebook operate enormous data centres, often distributed over various countries. These kinds of setups require a whole different type of configuration than simple databases to keep things running and working together. This configuration can furthermore also influence what information a user gets and how up to date it is. Because of this issue, just like with DNS resolution, websites can serve different content to different web bots that scrape from different locations. It is for the same reason an impacting factor.

Local Privacy Regulations and legislation: The internet stretches across the whole planet. All connected computers understand the same languages to transfer traffic. However, internet related regulations, such as privacy regulations, vary substantially across the world. Countries their local privacy regulations govern, for example, what regulations apply to the storage of personal information and the sharing (or selling) of this personal information with other parties. These regulations vary per country. In Europe, we have the GDPR regulations, in the USA, jobs and profit have a higher priority [MM12], so regulations are less restricted. Since tracking is so privacy intrusive, it gets scrutinised under the local privacy regulations. These regulations thus influence the tracking practices carried out in the corresponding countries. For example, Fruchter et al. [FMSB15] discovered that there is a significant difference in tracking activities per country. Privacy regulations do not only influence tracking practices, they can also govern advertising practices. For example, the injection of ads in internet traffic when an ISP passes it, is not allowed in the UK. Again, like with the other impacting factors in this category, scraping from different locations results in different web scraping results. This time it is related to user tracking. It is then also, for the same reasons, an impacting factor.

5.4. Impacting factors related to the server side

Dynamic web

Changes in content or layouts and updates: As we have mentioned different times, web content and the design of web pages change over time. Websites come and go; new web technologies introduce new layouts, updates and then again, some sites barely change their content, let alone their appearance. Other web pages, such as sites with blogs, change more frequently. And then, we have those types of websites, such as new sites, which change their content multiple times a day. Vissers et al. [VNBJ14] mention another example of the web's dynamism. Airline prices are highly volatile, and they change regularly for a variety of reasons. Social media websites additionally allow users to post their content online. Millions, if not billions, of users online daily make for a massive amount of new content. If a web bot (or another) scrapes the same website later, it might scrape different results because of the content updates. This also meets the definition of an impacting factor.

[4] https://www.youtube.com/watch?v=far015_0NUQ

Individualisation: Individualisation refers to those actions that allow servers to serve individual users personalised content. Servers can base this on the previous tracking or preconfigured settings. User tracking allows serving the user personalised advertisements. Some sites, for example, Youtube.com, serve their users similar video content based on what they viewed in the past. Again other sites, such as Facebook and Twitter, mostly show content posted by users which you selected to follow. Different web bots may scrape different content because of this individualisation. That is why this is an impacting factor.

A/B Testing: A/B testing refers to a type of research in which we can compare two variants of a website. In such kind of research, we create two groups and serve each group with a different website variant. On the web, these two groups would be website visitors. We can perform measurements on these two groups and later compare the results to determine which version performs the best. Common subjects for this type of research are ads, promotional materials and banners. If two web bots scrape the same website and this website happens to perform A/B testing, then these web bots might scrape different web content, which is why this is an impacting factor.

Cloaking: Wu and Davison [WD05] define cloaking as the practice of sending different content to a search engine than to regular visitors of a website. More general, Krumnow et al. [KJK22] defines it as the practice of delivering uniquely tailored content to specific clients, such as web bots. Websites can use it in benign ways [ITK+16], by, for example, redirecting mobile clients to pages optimised for small screens. But it is often used in more malicious ways, such as for search engine optimisation (SEO) [WSV11], to obtain user traffic illegitimately for scams. A cloaking technique that is able to detect a web bot but maybe not a stealthy web bot, will serve different content to each. That is what we understand under an impacting factor.

Errors

Server non-responsive: On the other side of our request lies the webserver. Although very robust to handle all kinds of errors, it cannot work without electricity. So any outages can cause severe troubles for the availability of the websites that the server hosts. Server owners have searched for alternatives and backups for the standard electricity net. An innovative idea is to use solar power to run the servers. Indeed today, whole server farms are powered by solar energy. Solar power provides an environment-friendly alternative to the expensive and polluting grid power. Facebook is an example of a company that planned to power its data centre with the help of solar energy [5]. Other causes for non-responsiveness are maintenance works. While administrators work on the critical infrastructure, certain services might not work. In a very rare situation, it could happen that one web bot performs a successful web scrape while a subsequent (or another web bot at a later time) makes a request to a server that is non-responsive. We consider this error then as an impacting factor for the same reasons as for the other errors.

[5] https://www.youtube.com/watch?v=_ZLqLWyCvHs

6

DEVELOPMENT OF A TAXONOMY

The development of our taxonomy involves the categorisation of the impacting factors that we identified. We, furthermore, want to limit our scope somehow, so we apply our model of a more generic scraping study that we discussed in the Research methodology chapter. With this, we aim to target those researchers who want to compare multiple crawls, whether they perform these with multiple web bots or a single one.

First, we will give an overview of the taxonomy, for which the development combined both the bottom-up and top-down approaches. Next, we present the different branches of our taxonomy and discuss our choices of categorising the terms.

6.1. LUMPING AND SPLITTING

Figure 4.2 shows our model of the scraping approach. In this model, we recognise three core concepts: client, server, and network. These concepts become the top-level concepts in our taxonomy. We split the whole taxonomy into these three key terms as a way of saying. An additional fourth major category consists of all those impacting factors that we choose as input parameters in the design of our scraping study.

On the other hand, for the impacting factors, we use the term lumping to indicate that we group them into categories.

Figure 6.1 shows the overview of the whole taxonomy. We highlight the top-level categories in blue. The four pictures after the overview show the different top-level categories (or branches), one by one, to make the terms easily readable.

The blue categories are the result of splitting, and we combine this approach with the opposite task. That is, we lump our impacting factors together. Again, we make this combination because it is the general consensus among different authors that write about taxonomies. The newly emerging categories also act as the meeting point between the two approaches that we use (top-down and bottom-up). In Figure 6.1 we give these new classes the colour green. The impacting factors have a yellow colour. To lump the related impacting factors together, we used a technique called card slots, which we borrowed from the book by Hedden [Hed10]. We basically write each impacting factor on a card and move them around to form groups (the categories). We write also write the categories on these card slots. We use this simple technique because of its benefits. For example, we can easily move the cards around and create new categories. Because there are many cards, it is easier to get an overview of which we categorised already.

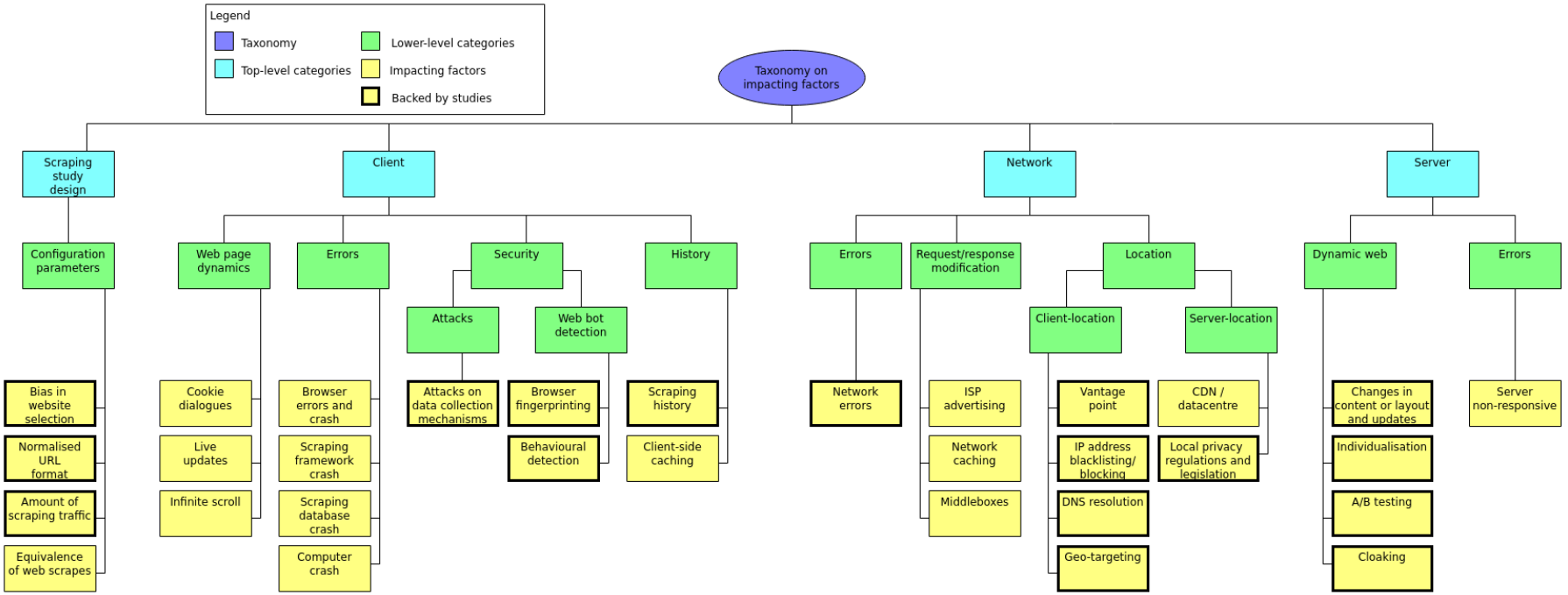

Figure 6.1: Taxonomy of impacting factors

6.2. TAXONOMY OVERVIEW

Now that we have our taxonomy, we can motivate our choices of categorising the different impacting factors into different categories. We group our discussions according to the four top-level categories from our taxonomy. These form four separate branches. One for the scraping design, one for the client side, on for the the server side, and the networking in general. For each of the top-level categories, we list the lower-level categories, briefly explain them, and discuss which impacting factors we categorise under them. The following picture shows a legend of the colours and accentuation that we use in the following figures.

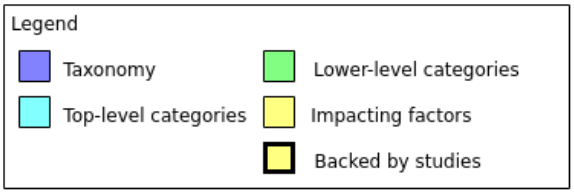

Figure 6.2: Legend of taxonomy categories and factors

SCRAPING STUDY DESIGN

We start with one category that we did not derive from the scraping model. Next to all the impacting factors that can influence our scraping results from the outside, there also exist factors that we completely control ourselves. When designing our scraping study, we need to make choices, such as what we will measure and how. We can control these aspects by the configuration settings of our scraping framework.

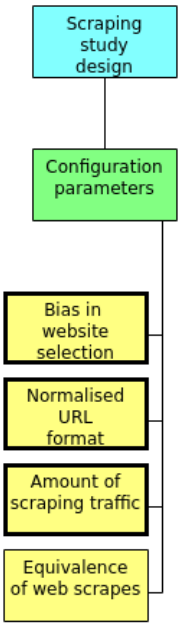

Figure 6.3: Client branch of taxonomy

- **Configuration parameters** For every scraping study, we configure some parameters so that we do not run into any trouble. Every study scrapes a selection of websites,

and we choose this as an input to the scraping run. Jueckstock et al. [JSS+21], for example, discuss their point of view about a random selection. Part of this input is how we write the URL format. For example, Zeber et al. [ZBO+20] choose to use a stem version. Finally, we can, in advance, choose to perform a certain number of scrapes, such as eight by Krumnow et al. [KJK22].

Client side

The client side comprises the systems and all the web bots that run on them. We can also think of any additional services that someone can run to collect and store the web scraping results. It encompasses all those factors that impact these systems, including any website peculiarities that appear on the client side after loading the web page.

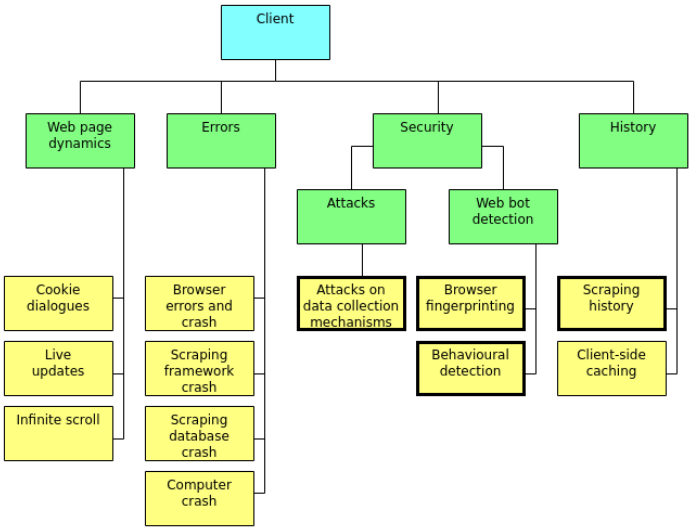

Figure 6.4: Client branch of taxonomy

- **Web page dynamics:** Web page dynamics refers to those factors that impact how the client perceives the web page after it is loaded. After some user actions, the web page can load additional content, which impacts the collected data.
- **Errors:** Errors can appear in any part of our scraping model. This category only includes those factors that are related to errors on the client side. We identified four general areas where the errors could occur. These are the browser that runs the web bot, any processes involved with the scraping framework and its use of a database, and the whole computer system in general.
- **Attacks:** Security issues are a constant cause of concern when surfing the web. Web bots make web requests, just like human users do through their browsers. So web bots can be exposed to the same threats as users. Now web bots are not so naive to fall for phishing scams attacks against users can also affect web bot operations. Researchers have not written much about attacks against web bots, but the work is ongoing.
- **Web bot detection:** Web bot detection targets specific client-side features. It is the client that is able to control some of those features. There are different techniques

to detect a web bot. We include the two major categories, which are browser fingerprinting [JKV19] and behavioural web bot detection [CGW18]. Including every minor technique within these two would go too far for this study.

- **History:** Some privacy threats, such as cookie synchronisation [PKM18] and price discrimination [MGEL12, HTWH18], can only be practically measured when you take previous web activity into account. So web surfing history, for example, or the lack of it, can impact cookie synchronisation measurements. History comprises any kind of surfing basis for further measurement that you can build up, such as a scraping profile, which is used by the study of Vissers et al. [VNBJ14].

NETWORK SIDE

All requests and responses travel through the network. With the networking side, we consider any factors related to the network that the scraping uses, that can have an impact on the returned results. We consider errors, any kind of manipulation and impacting factors related to the used protocols, such as the used IP address. The server side can derive the user's location from the IP address, which forms the basis of many impacting factors.

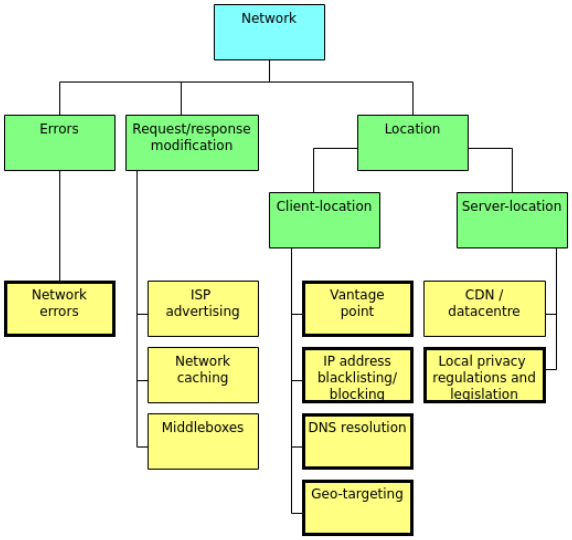

Figure 6.5: Network branch of taxonomy

- **Errors:** With such a huge and complex internet, we almost take it for granted that network errors occur. The HTTP protocol has an extensive list of HTTP status codes that notify us of any network-related errors. We mainly target the problem of network congestion, that many studies, such as that of Pham et al. [PSF16], report as an impacting factor.

- **Request/ response modification:** Not all deviating website responses are the result of errors or countermeasures by the server against web bots. ISPs, for example, as the man in the middle, saw the opportunity to inject ads into the user's internet traffic. Other technologies, also in the middle between client and server, can undeliberately impact the web content. We mention caching and middleboxes.

- **Client-location:** We use the category "client-location" to group all the factors related to the location from where the user surfs the web. The study by Jueckstock et al. [JSS+21] mentions vantage point, IP address blacklisting, and DNS resolution. Geo-targeting is a well-known phenomenon that is addressed by, among others, the study by Englehardt and Narayanan [EN16].

- **Server-location:** Not only the user's location can influence the scraping results. The location of the server's data centres or services with whom they collaborate can affect the results. This category includes all kinds of server-side infrastructure. More abstract things, such as the local privacy regulations of the server's location [FMSB15], also belong to this category.

SERVER SIDE

The server side includes all the impacting factors that the server can control. The server decides, for example, what content to serve and to who.

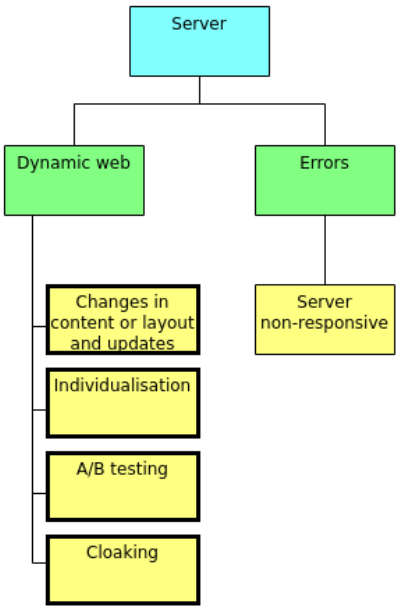

Figure 6.6: Server branch of taxonomy

- **Dynamic web:** The web's dynamism has many causes. All are the result of the server's initiative to update content [ADZ+20, PSF16], test user preferences with A/B testing [KJK22], cloaking [ITK+16] etc. The dynamic web category tries to comprise all these causes.

- **Errors:** The network handles all errors away from the client in the same way, with HTTP error codes. There is even a specific category of error codes for server errors. We still included a separate category for server-side errors because of the different nature of the errors.

7

ELIMINATING IMPACTING FACTORS

We will again list the confounding factors that we identified in Chapter 5. This time, however, we will look at the problems they may cause and ways to mitigate or manage the impact of these factors. Before we discuss every mitigation in detail, we want to give an overview of this chapter. In Appendix A we included a table of all the impacting factors and corresponding mitigations that we will discuss in this chapter. Furthermore, now that we presented our taxonomy, we have a better overview of all the impacting factors and how they relate to each other. We can also use the taxonomy to illustrate some additional data from our table. In Figure 7.1 we colour-coded each impacting factor with the extent to which we can mitigate it. From this figure, we can derive that roughly half of the impacting factors are fully mitigatable. We gave these impacting factors the colour green. For the other half, we can place a note on the actual effectiveness of the mitigations. We gave a mitigation for the orange-coloured impacting factors, but we have to take their use with a grain of salt. Finally, there exist two red-coloured impacting factors that we cannot truly mitigate.

Next follows a discussion of the problems and solutions for each of our impacting factors, grouped per branch in the taxonomy. Afterwards, we search for relations between the mitigations that we found for the impacting factors and the categories of these factors. We finally discuss some techniques that we can use to address the problem of identification and mitigation of still unknown factors.

7.1. SCRAPING STUDY DESIGN RELATED IMPACTING FACTORS

CONFIGURATION PARAMETERS

Bias in selection of websites

Problem: If we take, for example, a very naive website selection method, such as the first thousand websites. Then, when we derive conclusions from the scraping results, we can only make statements about a very small tracker set. From the study of Englehardt and Narayanan [EN16] we know that the largest trackers appear in the majority of sites, while there also exist a long tail of trackers that you can then miss in your research.

Solution: You need a website selection method that allows you to draw conclusions about all the websites in general. We also need a random selection technique to exclude any personal preference for specific websites. A solution is to divide a long list by the number of

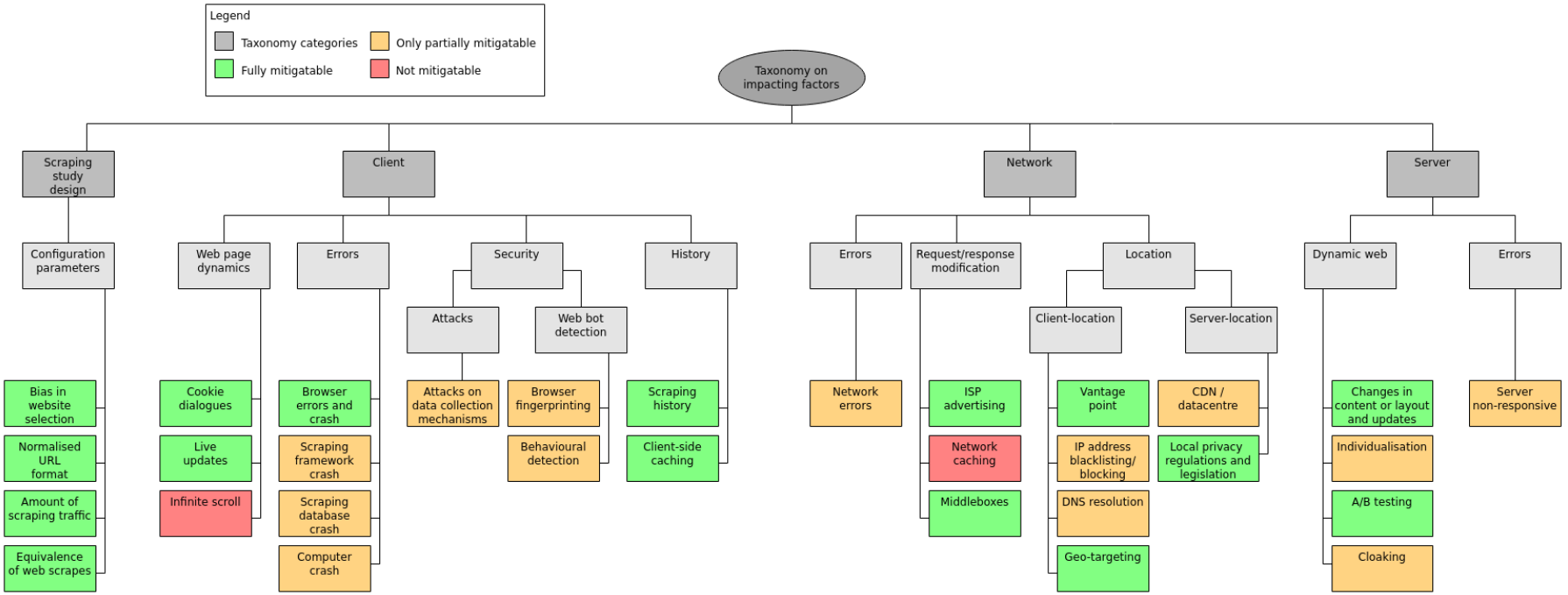

Figure 7.1: Colour-coded taxonomy shows which impacting factors are mitigatable

websites we want to scrape. From each slice, we will then select a random website. Krumnow et al. [KJK22] solve this problem by dividing the total number of websites (Tranco Top 10,000) in ten buckets, and then select one hundred random websites from each bucket. Jueckstock et al. [JSS+21] shuffle the website list that they use in advance. This shuffling decouples the website's popularity from the elapsed time, and it prevents any accidental detection.

Normalised URL format

Problem: Errors caused by a wrong URL format will not differ between scrapes; they will be specific to the website. The concern here is to avoid errors. Some websites only accept connections with the HTTP protocol; others require you to use the secure HTTP version (HTTPS). If you consistently use one type, it will cause problems for some websites that do not accept HTTPS connections. One of the errors can be that you get a *301 Moved Permanently* HTTP redirection response. OpenWPM can handle any redirects, but we want to avoid any errors in advance.

Solution: Zeber et al. [ZBO+20] also faced some problems with the URL format. They solved it by stemming the URL. Stemming basically means that we strip everything from the URL, except the hostname and path components. For the starting URL, this corresponds to the URLs in the format from the TRANCO list. So you can use them straight from that list.

Amount of scraping traffic

Problem: Exaggerated parallel scraping by multiple web bots could cause problems if a server engages in some form of rate limiting [VNBJ14]. The server could then perhaps reject requests from a specific web bot. The study by Pham et al. [PSF16] also avoids hitting a website multiple times to avoid web bot detection and, as a consequence, the blocking of their requests. You do not want a server to block you in advance or across multiple websites.

Solution: The study by Vissers et al. [VNBJ14] does not execute scrapes in parallel not to disrupt the server and prevent rejection after rate limiting. The study by Krumnow et al. [KJK22], on the other hand, uses eight scrapes per web bot. They find this suspicious number still an acceptable trade-off to establish a baseline for comparison. Three web scrapes per web bot are the minimum number to be able to account for any incidental deviations in one web scrape. You could furthermore use a VPN Proxy to route some scrapes through another IP address. This way, you can limit your footprint in case the server employs rate limiting.

Equivalence of web scrapes

Problem: Our generic scraping model involves the use of multiple web bots. It is almost too trivial to mention that if all web bots do something different, their results also differ. Furthermore, we expect that some web bots receive a different response from the webserver. These differences could manifest themselves in differences in user tracking, but, for example, as detected in the study by Vlot [Vlo18], different web content is also possible. And different web content most probably means different links. So if each web bot follows different links, they will consequently collect different data.

Solution: We thus want to add the constraint that all the web bots must measure exactly the same web pages so we can properly compare the tracking measurements. To do this, you need to explicitly define what the equivalence of web scrapes means for your project.

Then you must implement your scraping setup accordingly. For example, to not poison your measurements, all the web bots must scrape the same links. To accomplish this, you must choose which set of links you will use and, furthermore, you must share this set with all the web bots.

7.2. IMPACTING FACTORS RELATED TO THE CLIENT SIDE

WEB PAGE DYNAMICS

Cookie Dialogues

Problem: The cookie dialogue might block the web page that you are scraping. In case you wanted to scrape any content, this would block your efforts. After you selected your preferred settings, the website will remember them for your next visit. So your next visit, the cookie dialogue will not show up and you will measure different data. Does the website set any cookies before you accept them? If you perform a user tracking study that measures cookie-related issues, such as cookie synchronising, you probably want a website to set their cookies.

Solution: Regarding the problem that the dialogue blocks the web page. Using any automation framework, such as Selenium, you could easily click the agree on all button. However, this is not a scalable solution. There do exist solutions that handle the cookie dialogues. On example is a Firefox plugin, *I don't care about cookies* [1] , which removes cookie warnings. Some of these solutions employ pattern matching for known keywords, which has the disadvantage that they may miss some dialogues for some websites. Some studies reported that websites are also able to fingerprint plugins. In this case, your fingerprint would make you more unique and extra detectable, which is not really what you want.

Live updates

Problem: Live updates, such as getting the latest tweets in real-time, happen uniquely in time. You do not get the same live updates again. So this evidently does not pose any problems if you perform a single scrape. With multiple web bots that each perform multiple parallel scrapes, live updates pose a different challenge. Suppose that you perform the scrapes after each other or maybe, not at exactly the same time (possibly different page loading times). Web scrapes performed a bit later may include live updates that earlier web bots did not see. So, different web bots scrape different data, which may lead to different results, depending on what you measure.

Solution: A solution for the case of multiple web bots is twofold. First of all, you need to perform all scrapes, synchronised, at exactly the same time. This way, you all see the same live updates or not at all. Secondly, by performing multiple parallel scrapes per web bot, you can later filter out any accidental live update that only appeared to one web bot.

Infinite scroll

Problem: The infinite scroll that some websites implement allows loading additional content by scrolling down a little bit. Depending on whether you measure the full page content, this technique might pose a problem. In case you scroll down automatically, the page content only loads slowly or just incrementally. In these cases, you will only load a part of the content.

[1] https://addons.mozilla.org/en-US/firefox/addon/i-dont-care-about-cookies/

Solution: In case you really need to load the full page, you will need to code a method that performs this action. Loading more content takes time, and given the massive amount of content that some pages can load, this might add up. You will, furthermore, not get any signal that the final content was loaded. If you scrape thousands of websites this way without knowing which implement an infinite scroll, you are wasting a lot of time. Above that, scrolling by a web bot is easily detected by behavioural web bot detection. Performing long page scrolls will get you detected on some websites. Our advice would then be, not to perform such actions and settle for the initial page load.

ERRORS

Browser errors and crash

Problem: Despite the many advantages, OpenWPM does tend to crash once in a while during long scrapes [EN16]. This has consequences for the data collection, such as data loss. In a research project that deploys multiple web bots, this has even more disadvantages. If one browser instance from one specific web bot crashes, then you cannot compare the scraping results from all the web bots.

Solution: OpenWPM provides robustness by utilising isolated measurement processes and it has its build-in browser-crash recovery functionality [EN16]. However, if a browser instance crashes, then this will affect the symmetrical flow between the different web bots. Prevention is nearly impossible, so you can choose to log all success or failures from all the scrapes in a separate, easy to analyse (or easy to automatically process) log file. After the scraping runs finish, you can manually (or automatically) inspect the log to determine on which websites the failures occurred. You can then decide to omit the results for such websites or scrape them again.

Scraping framework crash

Problem: Your scraping framework hopefully never crashes, but if it does, then your whole scraping run halts. You can probably not recover from such a crash, so all further measurements seize. If you cannot scrape any further, then you can neither collect any data for your measurements.

Solution: We do not expect a scraping framework to crash, but you can never know. If it crashes, there is little that you can do about it. If you start the process from a Bash script, then you can monitor if the process exits with any code other than the normal code. You can then log success or failure. You will need to restart the whole scraping run to ensure that you did not miss any data.

Scraping database crash

Problem: The scraping database can crash if its file (or file content) gets corrupted. Whatever the reason for this, a crash will stop the data collection, be it a deliberate assault or programming error in the scraping framework. Then you get the same problem as with a scraping framework crash.

Solution: A solution to a database crash depends on the anomaly that caused it. It is better to start your search, beginning with the problem that caused the crash. You can additionally log and monitor your progress so that when a crash occurs, and you are able to recover the previous data, then you do not have to scrape the part that went fine again.

Whole computer system slowdown or crash

Problem: System crashes would ruin the whole scraping run. In the absolute worst case, slow systems could slow our scraping down so much that the processes time out every time.

Solution: These are the extreme exceptions, which we do not expect to happen. One possible solution is to monitor the progress manually and terminate the scraping process if something appears very anomalous. The study by Jueckstock et al. [JSS+21] sets CPU, memory and bandwidth limits on all workers to lock total use of the system resource below the potential saturation for the lowest common denominator environment.

ATTACKS

Attacks on OpenWPM data collection mechanism

Problem: The problem with security issues is that they break the program's functionality. In this case, an exploitation of the vulnerabilities, found by Krumnow et al.[KJK22], could block the data collection mechanism, allows to inject fake data, or attack the completeness of the collected data. These vulnerabilities concern specific types of data, such as, the HTTP traffic, the JavaScript, and the cookie data. Since collecting data is the main purpose of using OpenWPM, these vulnerabilities could pose a threat. This problem sounds grave, but on the other hand, the research paper is not published yet, so the vulnerabilities are not widely known. Furthermore, it can only be exploited by the website that you scrape. So one website at a time could attack you.

Solution: Krumnow et al.[KJK22] propose some solutions for part of the problems. To prevent the blocking and injection of fake data, you could instrument the setter of the event dispatcher (with was the vulnerability). This solution would make any tampering traceable. For the attack on the completeness, Krumnow et al. propose to enable full HTTP recording.

WEB BOT DETECTION

Browser Fingerprinting

Problem: Websites deploy web bot detection mechanisms, such as browser fingerprinting, because they perceive web bots as an intrusion. Detection alone do not stop web bots, so some websites do also put mitigation mechanisms in place, such as rate limiting. Vlot [Vlo18] identified three other categories of deviating website responses after web bot detection. These are the page blocking, blocking of specific content (for example, high bandwidth content, such as videos), and showing different content. If these differences are relevant to your measurements, then web bot detection will probably negatively affect your scraping results and study.

Solution: One obvious solution to finding whether you got a different website response, is, to compare the response of a web bot against one where you manually visit the same website. This solution is however not workable if you plan to scrape large numbers of websites. The closest solution to resemble a human visitor is to use a less detectable (stealthy) web bot. Jueckstock et al. [JSS+21] and Krumnow et al. [KJK22] both carried out a privacy-related study where they compared there web bot results against a less-detectable web bot. Both studies show that websites respond differently to the more detectable web bot.

Behavioural web bot detection

Problem: The web page collects user events, so problems start when you program your web bot to perform any page actions. It could also be possible that not performing any actions appears suspicious to the web server. However, we assume here, that behavioural web bot detection only happens after a certain amount of user events. Detectable behaviour of web bots are for example, automatically typing in a form field and moving the cursor around to certain objects and clicking them. We assume that any scraping strategy that includes page interactions could lead to the detection of the web bot, with all the consequences that may follow.

Solution: The easiest and safest solution would be, to not perform any interactions. In case that you absolutely need to perform any actions, such as scrolling or moving to an item to click it, then you must try to model it as human-like as possible. You should avoid straight lines, constant movements and speeds, continuous scrolling etc. OpenWPM itself offers some build-in basic function that performs human-like behaviour. Krumnow et al. [KJK22] further indicate that a few extensions would allow to make any web bot interactions less distinctive from human behaviour.

History

Scraping history

Problem: Web bot scraping history, or the accumulation of data from the set of websites that you previously visited is not necessary for most web measurements. In fact, keeping such data may allow other websites to derive your scraping history from it. It is very well possible that this history, now in the hands of the server, influences the response data. For example, you get other types of cookies. Obviously, this can affect the research results.

Solution: There is an easy to apply solution if you are using the OpenWPM framework. It allows you to configure whether you want to perform statefull or stateless web crawls. This corresponds to keeping the history while you scrape or deleting after every website scrape. Englehardt and Narayanan [EN16], for example, performed a one million website crawl, using the stateless configuration.

Client-side caching

Problem: Caching cannot pose a problem if we scrape every website a single time. In our scraping model, however, we perform multiple scrapes by multiple web bots. So if we perform all these scrapes from a single computer, then we will certainly witness some performance benefits. On the other hand, the use of caching will give the wrong results in some cases where the websites update frequently. Then we might miss the latest data because the cache serves us old content.

Solution: You cannot continuously clear the cache when you perform the web scrapes. It is possible to do it after the scraping runs. Besides that, you should limit the number of web bots per web cache to one. This way, the cached content cannot interfere with the requests of the other web bots.

7.3. NETWORKING-RELATED IMPACTING FACTORS

ERRORS

Network errors

Problem: We are mainly talking about HTTP errors. All HTTP responses and errors carry an HTTP response status code. Other network problems are less direct. For example, network congestion may delay our request for a website. If congestion is too high at a certain time, it will result in a timeout limit and, consequently, a failed scrape. Such errors might cause deviations in the data we collect (we will most probably collect less data). When we later analyse the data and compare them to the results of the other web bots, we will derive the wrong conclusions.

Solution: We cannot prevent all network errors. What we can do is, log them, monitor the logs, and take appropriate actions. For example, we can still try to prevent timeout errors by allowing more waiting time to load the page completely. The study by Zeber et al. [ZBO+20] uses the shortest timeout, of ten seconds, of all reviewed studies. Ruohonen and Leppänen [RL18] furthermore process their website list three times to rule out any temporary network failures. The study by Pham et al. [PSF16] uses a delay of twenty minutes between, maximally five retries, to avoid detection by websites that look at request frequency.

REQUEST/ RESPONSE MODIFICATION

ISP advertising

Problem: How can we know whether the ads that we see on our web pages are those from the advertiser or from our ISP? If there further is no connection between requests and responses for the advertising content, then any studies on ads will draw false conclusions. Another problem is, that the ISP serves personalised ads. So there are no fixed ads for everybody. Perhaps that they rotate a series of standard ads, but we cannot know that. When we perform parallel web scrapes, every scrape may get different ads. These results may be confusing or even troublesome if we research ad related subjects.

Solution: There are a number of solutions online, posted as a response to complaints about this shady practice. The most straightforward solution, is to install an ad blocker [2]. Another solution is to use a VPN proxy to a proxy outside of your ISP's control. This solution will hide your traffic from your ISP, which will prevent him from injecting packets. Finally, you could choose to change your ISP. A lot of complaints and losing customers might push your ISP to drop these shady practices.

Network caching

Problem: The problems of network caching resemble much with those of client-side caching. Cached content may influence the results of other web bots or web scrapes. In this case, it can be the results of all web bots connected to the same network (we mostly mean an enterprise network). It might also be more difficult to get access to this type of cache to clear it.

Solution: Since network caching is mostly associated with enterprise networks, and since we know that these networks are easily recognised, and their IP address gets blacklisted, we would advise not to perform any web scrapes from within such networks. Fur-

[2] https://superuser.com/questions/902635/isp-is-inserting-ads-into-web-pages

thermore, there is no other straightforward solution since it can be difficult or inappropriate to access the network cache.

Middleboxes

Problem: Although middleboxes might perform some important functions within your network, these functions might interfere with the web bots' scraping traffic. We, for example, know that older types of middleboxes are incompatible with new protocols, such as HTTP 2.0. Above that, some of their functions need to decrypt TLS traffic to be able to route the traffic correctly. This action violates important architectural principles, such as the end-to-end principle.

Solution: Although there is a higher chance that enterprise networks use middleboxes that have a greater impact on the web scraping results, there also exist middlebox functions in standard private network devices. So, whenever you start scraping, you must first determine what kind of middleboxes are present in your network and what their impact may be. Since private residential networks, with or without middlebox features, are the most (normal user-like) realistic baseline, you can compare your situation against your home network. If there is a lot of difference, then you should avoid scraping from such a network. The safest option is to use a private residential network.

CLIENT-LOCATION

Vantage point

Problem: From the website owners' viewpoint, the IP address can indicate from where you scrape, or, in other words, which vantage point you use. One way to deal with scraping web bots is to blacklist them. The IP address types that get easily blacklisted are those of companies and universities. University subnets of IP addresses are assigned within special ranges, making them suspicious as web bots sources. IP addresses of scraping proxies are obviously also blacklisted. For residential IP addresses, it is a different story. Residential IP addresses belong to innocent civilians. Blacklisting them can lead to mistakes and, further on, a bad reputation.

Solution: The solution is to use residential IP addresses, which resemble the closest to a regular human website visitor. This lessens the chance that you get blacklisted and deviating content responses. You can also use a VPN Proxy to hide your IP address, but a VPN proxy's IP address can also get blacklisted. As a solution, some internet service offers another type of proxy service: Rotating Backconnect Proxies [3]. Backconnect means that the service uses a single node to allow you to connect to a pool of residential proxies. Rotating means that you acquire a new proxy for every request you make or acquire one every thirty minutes or so.

IP address blacklisting or blocking

Problem: Ahmad et al. [ADZ+20] write, that, as a consequence of a web server linking our web bots with our IP address, it might trigger IP address based discrimination. Because the server will always get to know our IP address, he might link it to a malicious web bot activity. Blacklisting can manifest itself in, for example, a *403* HTTP response status code or a CAPTCHA challenge. Furthermore, the study by Krumnow et al. [KJK22] shows that when

[3] https://smartproxy.com/proxies/backconnect-proxies

one scrape gets blocked, then it is also likely that all the other scrapes under that same IP address also get blocked. If we get blacklisted, this will introduce a deviations in our results.

Solution: Our main solution relies on the use of a VPN proxy. The use of a proxy will make it harder to detect your private IP address. It is, however, not a perfect solution, because internet services, such as ipinfo.io [4], can detect whether you are using a VPN Proxy. Alternatively, you can route your requests through a TOR network, as in the study by Pham et al. [PSF16]. It will hide your true IP address but a website can still detect that our request came through this TOR network. We can find out if our IP address got blacklisted and possibly gets discriminated. Both the study by Jueckstock et al. [JSS+21] [5] and that by Ahmad et al. [ADZ+20] [6] consult online services that check a range of IP blacklist lists to check if their used IP addresses got a bad reputation.

DNS resolution

Problem: ISPs assign you a DNS server in your proximity to not overburden a single central DNS server. The use of a DNS server in your proximity may leak some information about your location. If the website that you scrape, employs some special mechanisms, they may be able to even pinpoint your exact location [7]. For the cases of distributed internet services, such as cloud services and Content Delivery Networks, different DNS servers may furthermore also resolve URLs to the IP address of those services' servers in the user's proximity. So, if you are scraping from different places, you might get redirected to different servers, that possibly store different versions of the data that you need.

Solution: We can address the problem of leaking our location, by using a VPN proxy. The proxy will hide which website we visit, but it will also mask the DNS server that the website can see. Next we have the problem that a request for a URL will resolve to different IP addresses, depending on the location of the request. Jueckstock et al. [JSS+21] try to work around this problem by using a special DNS resolver service. They were able to proxy their DNS requests to this service from their Chrome-based web bots.

Geo-targeting

Problem: The problem with the influence of geo-targeting is that different web bots might get different website responses, depending on their location. Ahmad et al. [ADZ+20], for example, note that web servers might return localised results, which means that in different places in the world, you can get different website responses. They further argue that localised results make it challenging to rely on cloud infrastructure for measurements. In rare cases, a foreign power can block access to certain websites.

Solution: In case that we use different web bots, we should deploy them from approximately the same location. If this is not possible, because for example, you use a cloud service, then you can then try to counter any negative influences by choosing a VPN proxy, for the web bots in different locations. You can choose a proxy in the same country as your residential IP address, or a neighbouring country, but still the same geo-political region.

[4] https://ipinfo.io/
[5] https://hetrixtools.com/blacklist-check/
[6] https://talosintelligence.com/
[7] https://oxylabs.io/blog/what-is-an-ip-address

SERVER-LOCATION

Local Privacy Regulations and legislation

Problem: Many websites, such as those of international retail webshops, have local versions of their websites. You get such a local website if you scrape from a location in the proximity of that country. For example, if you use multiple web bots from multiple locations, then one web bot could get a different local version. Different local versions may advertise other products, but the local privacy regulations may be different too. These differing regulations might have an influence on the website's tracking practices. And different tracking practices means different scraping results.

Solution: Obviously, the problem lies with the location from where you deploy your web bots. The solution requires us to perform the web scrapes from the same approximate location. We can physically scrape from the same (approximate) location or use a VPN Proxy to position our web bots within the same geographical location.

Content Delivery Network/ datacentre

Problem: Very complex backend database architectures is one thing. Another thing is, keeping the whole system synchronised in this era of a super dynamic internet. How fast do you think you can synchronise the number of thumbs up on a Youtube video or the number of likes of Facebook posts on a worldwide level? Bad synchronisation may lead to deviating results among different web bots that scrape from different locations. The same applies to CDNs. If they host dynamic content, then the content provider must continuously update this content. An update to a legacy CDN can lead to latencies of up to one hour. When we employ a web scraper (web bot), we also get redirected to these CDNs for some of the web page's content that we requested. So our web bot's request gets redirected to the nearest CDN server, which might differ for the different web bots that we use.

Solution: Since content providers get so many benefits from using CDNs, their use becomes so widespread that we cannot avoid any requests to them. Given the problem of hitting different cache content, we need to make sure that our (proxy) locations remain close to each other. We could perform an extra check to verify that we do not get different data by performing multiple parallel and synchronised scrapes. So, we need web bots in the same geographical area, and we can combine this with our solution for the problem of different content after web bot detection (parallel and synchronised scrapes).

7.4. IMPACTING FACTORS RELATED TO THE SERVER SIDE

DYNAMIC WEB

Changes in content or layouts and updates

Problem: We all, of course, like the dynamic web with its fancy looks and exciting content. Ahmad et al. [ADZ+20] and Pham et al. [PSF16] previously already noted that crawls that start at different times, may observe different content because of frequent content updates by websites. Different content can also result in different tracking behaviour. For example, the web browser sends a request to another third party. And further on, other third parties may include additional content or ads.

Solution: We want to scrape, as much as possible, precisely the same web pages with each web bot. The study by Fouad et al. [FBLS20] reverts to synchronisation to account for the web's dynamism. They synchronise two statefull scrapes on two different computers

with a different IP address. To synchronise a list of websites, we have to take the different processing times of the different web bots into account. When these differences add up, then synchronisation can get disrupted to even one half an hour, as reported by the study by Jueckstock et al. [JSS+21]. Because of that, the study by Ahmad et al. [ADZ+20] uses a master script that coordinates the start of the crawling of a specific web page at approximately the same time. The study by Krumnow et al. [KJK22] account for this by synchronising the websites within one web bot per website and synchronising between the web bots per batch of websites.

Individualisation

Problem: In our scraping study, we do not have a lengthy browsing history with set preferences or a profile based on tracking. (Unless individualisation is based on the IP address, and by coincidence, our IP address gets detected.) So individualisation will start with a blank page (or history). Still, possibly, a website can serve random content and will act upon user preferences to later individualise it. In this case, the server will serve different web bots with different content, affecting user tracking.

Solution: You can try to mitigate such problems by scraping, with all web bots, as generic as possible and as similar as possible to each other. For this, you can, just like in the study by Jueckstock et al. [JSS+21], homogenise all controllable aspects of the crawls across all web bots.

A/B Testing

Problem: A/B testing could cause a problem for if a web server served one web bot, one version of the website and another web bot the other version. Another website version could further also result in different tracking measurements. If you study, for example, user tracking, and any differences result from A/B testing, then this would poison your measurements.

Solution: Our proposed solution against A/B testing is the same as that for individualisation and incidental data. You can follow the method from the study by Jueckstock et al. [JSS+21], which performs three parallel scrapes per web bot and provides for a measurement of stability across crawls. If A/B testing affects one of these scrapes, then the comparison of the parallel scrapes should filter this out.

Cloaking

Problem: if the cloaking website starts serving every crawler different content, then that may affect your study. In this case, the website could serve policy-abiding content exclusively to crawlers [ITK+16]. Normal human visitors would at the same time be exposed to attacks, or to user tracking (in our context). For example, when you try to study advertising scams, you may find that your crawler sees a benign page while human visitors see spam. Cloaking could possibly also target OpenWPM in particular [KJK22]. A website could detect OpenWPM through its fingerprint and then evade OpenWPM's data collection to hide its malicious practices..

Solution: A solution to detect cloaking [WD05], would be to calculate whether there is a difference between a copy from a search engine's perspective and a copy from a web browser's perspective. We thus need to compare a web bot crawl to the results of a human visiting the site with a browser. This solution, however, does not scale very well. A scalable

solution requires a less detectable scraper. So another possible mitigation is [KJK22], to hide OpenWPM's identifiable properties and make it less detectable.

ERRORS

Server non-responsive

Problem: The problem is plain and simple. If the server goes down, then we do not get any scraping results. The server can go down for many reasons. We are here mainly discussing power outages but actually, we witness the same response if there occurs an unrecoverable error. The alternative sources that we mentioned, such as solar power, can also fail. A few cloudy days and a high energy consuming server or data center, and your systems are down in no time. So you certainly need a backup power supply, and above that, you better backup your backup just to be sure.

Solution: We, the client side, we cannot control the server, let alone control any power failure that the server owner cannot even control. Our solution then lies in detecting whether there is a problem at the server side. The HTTP protocol can help us with that. Whenever there is a problem at the server side, the HTTP protocol will return one of the *5xx* series HTTP response status code. The most specific is the *503: Service unavailable* HTTP status code. The Mozilla documentation for HTTP status codes [8] explains that this means that *"The server is not ready to handle the request"*. This code includes that the server could be overloaded, currently down for maintenance, or just that the server is down. We should monitor the HTTP data of our scraping results for any signs of such HTTP response codes.

7.5. RELATIONS BETWEEN TAXONOMY CATEGORIES AND THEIR MITIGATIONS

Now that we developed a taxonomy and discussed the mitigations for our impacting factors, we can look at the taxonomy in the light of the mitigations that relate to the impacting factors. More specifically, we are interested in the relation between the mitigations within the different categories. This section will look at which lessons we can draw from the category-wise analysis of the mitigations.

If we take a look at the taxonomy overview in figure 6.1, we can distinguish a few types based on the number of impacting factors it contains. The number of impacting factors generally correspond to the number of mitigations within the categories. We, first of all, have those categories that contain only one impacting factor. We cannot say much about this type of category since there are no patterns to discover. Perhaps any mitigations related to the impacting factors here can also relate to the yet to discover unknown factors.

Much more interesting are the categories that contain multiple impacting factors. The first such category in the Scraping study design branch deals with the configuration parameters of the scraping setup. We can configure the corresponding mitigations before performing the scrapes. Although the mitigations vary completely, we generally see the tendency to keep the study more uniform and less detectable.

At the client side, we can conclude, for the web page dynamics category, that we want to generalise our scraping approach to make up for any keep out any scraping method that applies to only a specific type of website. We can, in general, do this by performing the web

[8] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#server_error_responses

scraping as generic as possible so that we can scale out. For example, we cannot deal with cookie dialogues by using a selection of different methods to accept cookies. This approach would most likely not work on several websites. So, we choose a generic approach, such as a plugin or ignore it altogether. We consider the same approach, with infinite scroll. We know a few websites where this applies, but if we scrape a huge list of websites, then we will certainly miss some (in case we use a list with known sites). So, again, we ignore the scrolling possibility to scrape all sites as equal as possible. We finally have the web bot detection category. The overall theme with the mitigations is to reduce our web bot footprint towards the server. For example, we can use a stealthy plugin. We further try to stay under the radar by limiting any page actions. We skip errors for now.

In the Network branch, we have two location related categories. Location impacts both the client side and the server side. This location mostly depends on their position within the network. Because of this, we categorise it under the network branch. It is very interesting to see that we can mitigate all the impacting factors (six of them) with the same technique, a VPN Proxy. The goal in each case is slightly different, though. With the vantage point factor, we are looking to use a specific type of IP address, although the location can also be an important factor. With IP address blacklisting or blocking, we are trying to hide our IP address. DNS resolution and geo-targeting are mostly location related, which we can both derive from the IP address. For the impacting factors of the server-location category, we can also use a VPN Proxy, but the emphasis lies more on scraping from the same geographical area than just changing the location.

We have some things to say about the Server branch. The dynamic web category actually comprises a part of the scraping approach, on which we based our scraping model. We perform multiple parallel and synchronised scrapes with multiple web bots. All mitigations in this category relate to performing multiple web scrapes in a synchronised manner. We later compare the scraping results to discard any incidental deviations. The model, in this way, also tries to deal with the more general impacting factor of the dynamic web.

We left the error category for last because we can find one such category in each branch. We can thus see it as a bit of a higher level category. Remarkably, there is also an overall way of mitigating the problems, regardless of the branch. Any solution lies in monitoring the web scraping progress and detecting an error or anomaly in real-time or in any logs.

7.6. Handling unknown factors

The development of our taxonomy ran against some time constraints and surely against our human limitations to process all available information. So there is a big chance that we did not identify every possible impacting factor that exists out there. We will refer to these unidentified factors as the unknown factors. Since they are unknown, we cannot say how many there are. If you design your study, you may run up to some new factors, or you may want to search for factors relevant to your specific study.

This section can help you further by giving a few tips and advice on how to search for unknown factors and related mitigations.

Identification of unknown factors

Before we discuss any mitigations, we first want to find those unknown factors.

The first and most obvious way to find unknown factors is to continue what we did to

find them. That is, to read and analyse more scraping studies. In case they treat another subject, they may have already identified any unseen confounding variables. But studies that treat the same subject may also mention new factors. It probably all depends on how advanced and recent the studies are. Researchers in more advanced studies have more expertise and know-how and therefore may identify new factors. While more recent studies build on past knowledge and may present new insights into older factors and present more specialised factors or sub-factors.

The next option is to start with the taxonomy presented in this thesis. A common technique to extend a taxonomy is to go through it, step by step, to see if you can think of any new terms. For this, you start at the top-level category. You then pass each category underneath it to see if you can find a new category or a new term that the taxonomy does not contain and is not included in any other term and category. You need to traverse the taxonomy in the same way if you already found any term, for example, through the process in the previous point. If you need to add a category for your new factor, it may be useful to think through that category to see if you can find any extra factors.

A handful tool that we used to search for any impacting factors was the model of a generic scraping study. It may be worthwhile to go through this model yourself to see if you can think of any new impacting factors because perhaps the context of your study gives you any new ideas. Finally, your study may be so different that you need to design a totally new model. Now that surely will churn up a load of new impacting factors.

In this discussion so far, we are dealing with the fact that the factors are unknown, and we cannot say in advance how many there are. This fact raises questions about to what extent that we can find new factors. We already indicated that this partly depends on the past experience and know-how of the researcher. We also mentioned that more recent studies could build on previous studies. Moreover, if researchers can team up, they have a better chance of specific factors that can impact a study. Furthermore, if the researchers can think through their study in more detail, then this may reveal new dependencies on other factors. Also, the model of the study can be more or less specific. Brainstorming over a more specific model can be more rewarding.

Mitigation of unknown factors

The whole point of searching for unknown factors is to try to mitigate them. The following chapter discusses the mitigations in detail. Here we will discuss the mitigations for the unknown factors.

If you did find a new impacting factor through one of the options that we described, you surely also want to mitigate any of the problems that it may cause. Again we propose some options that may be helpful to you.

The first option is, simply put, you can search for it yourself. For us, here, right now, it is just an abstract unknown factor, but you will have more information available. For example, you will know the specific impacting factor, the context where you found it, and you probably also already have an idea about the problems that it may cause. You can now search for a more meaningful solution than what we can say about it.

If we take a step back now and look at it in a more general sense, then let us take a look at the category from this new impacting factor (or unknown factor). Maybe, as we saw in the previous section, we can derive a specific mitigation technique from this category? These finding means that if your newly discovered unknown factor falls within such a type

of category, then this can give you a clue as to where to look for when searching for mitigations. For example, the category IP address contains multiple impacting factors from which the mitigation involves using a VPN Proxy. If your newly discovered unknown factor happens to fall into this category, then you should consider whether a VPN Proxy would also, somehow, mitigate your factor. The mitigations in a specific category maybe do not match at first sight. If you nevertheless abstract away a bit, then perhaps you can still find a common line in them.

FROM MITIGATIONS TO TYPES OF FACTORS

After we found an effective mitigation; we can also ask ourselves what else we could do with it. Can we use it to mitigate other factors? We can assign a greater value to mitigations that apply to multiple impacting factors (even if they are unrelated). This way, when designing a study and there appears a conflict between the mitigations of different impacting factors, we have an additional reason to choose a specific solution above another. On the other hand, if we now find additional ways to mitigate a factor, then we have more options. Or we apply multiple mitigations, thereby mitigating the impacting factor more effectively. Or we just choose our mitigation, when for example, one mitigation would not be effective at all.

We came up with a small process that consists of a few easy to follow steps. It allows you to search for those factors to which a mitigation applies. Here is an example of steps that we can use to determine which types of factors are affected by a given mitigation.

1. Take a certain mitigation.
2. Look at its impacting factor and the category of this impacting factor.
3. Try to come up with keywords associated with the mitigation. For example, for the use of a VPN proxy, we can think of keywords, such as location, IP address, security, privacy etc.
4. Can you lay a link with the other categories and impacting factors in the taxonomy?
5. Check if the mitigation affects the other impacting factors and those within this category (or categories).

8

How to quantify the impact of factors and mitigations on the measurements

After we found and categorised our impacting factors and their mitigations, we also want to know to what degree they impact our measurements. This way, you can determine whether they affect your study and how much.

So, we want to know their impact on our measurements. We can determine their impact, in general, through quantifying the differences between web scrapes. The quantification methods partly also depend on the kind of measurements that you quantify. For example, additive measurements (numeric) undoubtedly differ from more qualitative measurements, such as visual content. Besides this, we see differences, but also similarities, in the kind of quantification of different impacting factors.

The final goal of this chapter is to propose a starting point of a generally applicable quantification method to measure the impact of our factors. This chapter's title suggests that we also evaluate quantification methods for mitigations. If we logically reason about this, then we can see that any quantification of a mitigation is, in fact, the same as the quantification of the impacting factor. Contrary to measuring large differences from the impacting factors, we want to measure only minor differences in comparison with a baseline for the mitigations. Because of the similarities, we will limit our discussion to the impacting factors.

We do not have to develop a whole quantification system for all kinds of measurements from scratch because there already exist studies to base ourselves on. Our research includes a handful of studies that researched the influence of specific impacting factors. It would go too far to start a discussion on every impacting factor of our research. However, we will discuss how far we can generalise the quantification methods.

Before we evaluate any quantification methods, we will discuss a few general categories of impacting factors that differ in the way in which we would quantify any impact. Next, we assess a selection of quantification methods that we found in our literature. We discussed the context of the study in the Related work chapter. Here, we will briefly discuss what and how they measure (or a selection of them), and a quick summary of the findings based on the given quantifications. We, afterwards evaluate the given methods based on the points that we discussed in the Research methodology. We finally conclude this chapter with a recommendation of the most appropriate quantification technique.

Differences of quantification between categories

As we already mentioned, we cannot quantify the difference of all impacting factors in the same way. However, there are also similarities between the impacting factors. So we can categorise them. Firstly, according to the quantification method but, there are also impacting factors that we do not want to quantify. We limit the scope of our work by excluding these from our further evaluation.

Figure 8.1, on the next page, shows our taxonomy with colour coding for the impacting factors. We assigned the colours yellow and green to impacting factors that we can quantify in one way or another. For the orange coloured impacting factors, there are some barriers to be able to measure any impact in a straightforward manner, so we will not quantify them. Finally, in red, the factors that we want to exclude from our discussion.

We will shortly discuss some examples of the (coloured) categories.

We start with the red factors, which include all the error related impacting factors. In different studies, we noted that they exclude the data that include errors. So you can count errors, but omitted from the final results, we do not measure any impact. Then we also have two impacting factors (amount of scraping traffic and IP blacklisting and blocking). We will always want to avoid any related consequences (more certainly if they would be permanent), so we would certainly also not want to trigger them for quantification.

The orange impacting factors include most of those from the scraping study design category. If you choose differences for multiple web scrapes within these factors, the study changes so much that you would compare apples with oranges. Another example is client-side caching. If you want to measure any impact, you should compare one set of web scrapes that uses the cache and one where the cache is emptied. But this would mean that you would visit the same websites at different times, which raises questions about the influence of content updates etc. Because of these kind of barriers, we would need to tailor any quantification method to the individual impacting factors. In this chapter we limit ourselves to the most generally applicable methods.

The yellow and green impacting factors are those factors from which we can easily compare the scraping results. The difference between the two is that, for the yellow, we in general only compare the scraping results from two web scrapes. For the green, on the other hand, we want to compare multiple web scrapes (more than two). For example, if you want to measure the impact of a location-related impacting factor, then you would get more meaningful results if you performed web scrapes from more different locations. In contrast, for scraping history, you compare web scraping results that include or exclude the browsing history, so we compare two web scrapes. We will base the following section on these two categories.

8.1. Most suitable quantification methods

We will now discuss a small selection of quantification methods that we identified in our literature. We will start with the simplest form, plain tabular data. Next, we will focus on more graphical ways to organise the data. These methods should allow us to make conclusions easier. We finish with our proposal.

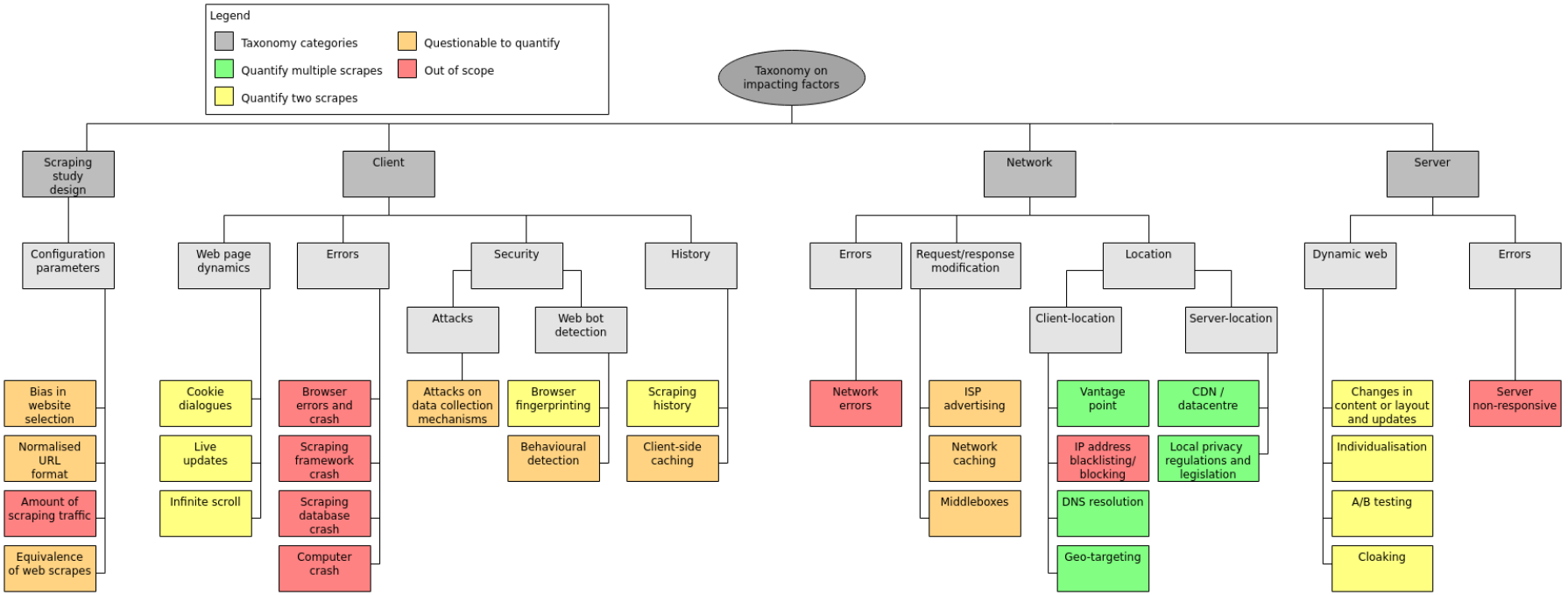

Figure 8.1: Overview of the different types of quantifications of the impacting factors

Local Privacy Regulations and legislation

Location was an important category within our taxonomy. The location of the server within the network is also an important factor, as the study by Fruchter et al. [FMSB15] concluded. Their research limited the number of countries to four, the United States of America (USA), Japan, Germany, and Australia.

On the visits to a selection of websites from these different locations, the authors measured the number of third-party HTTP requests and third-party cookie domains related to tracking and advertising. The authors used a statistical model that results in a ranking of the different measurements.

They present their ranked results in tables. You can immediately see that there are more third-party HTTP requests for the USA, and so, there is more tracking from that country. Other conclusions drawn from the differences between the countries' data, using the tabular results, requires pair wise comparisons. Because of the limited sample of countries, the authors could not draw real conclusions about the regulatory models.

Evaluation: The statistical model is very suitable to compare data groups, such as tracking data per country, in this case. They also use other statistical figures to reinforce their claims, such as a Chi-Square Statistic (χ^2), p-value, and Degrees of freedom (df).

The results are presented numerically in a tabular form which allows us to understand the figures in detail. The principle of rankings is very easy to grasp, but it is not the most appropriate kind of outcome to measure differences. However, you can easily calculate the difference between ranks and numbers and, possibly, present these results in the form of percentages (but this requires additional calculations). Besides this, for four countries, we can easily compare the data between each other. But if we use a lot more data (countries), then this comparison might become tedious.

Since the study by Fruchter et al. relates to the location category in our taxonomy, we can easily apply the method to all other factors within this category. Evidently, we can also present two categories in a table, so this quantification technique is generally applicable. We note here that all data were aggregated according to each location, so we can only draw conclusions for the data as a whole.

Vantage Point

With vantage points, we want to quantify the impact of the specific type of network type on the scraping results. The study by Jueckstock et al. [JSS+21] lends itself very well for this task because it already quantifies its measurements for different combinations of the vantage point. For this discussion, we can limit ourselves to the measurements of the volume of HTTP requests to third-party domains.

They use the term "bias score" for the logarithmic scale to quantify the differences between two scrapes. It divides an axis into a positive and a negative side. A tendency for one side would indicate that one scraping setup combination (one side of the figure) makes more HTTP requests to third parties. The resulting numbers used in their quantifications are the median of the three parallel web scrapes that they performed. This way, they only keep those measurements that are consistent across all three web scrapes.

In Figure 8.2 you see the type of graph, which is called a population pyramid, that the study by Jueckstock et al. use. On the horizontal axis, we see the bias score. Left and right from the central zero column, you have two results from the two different scraping con-

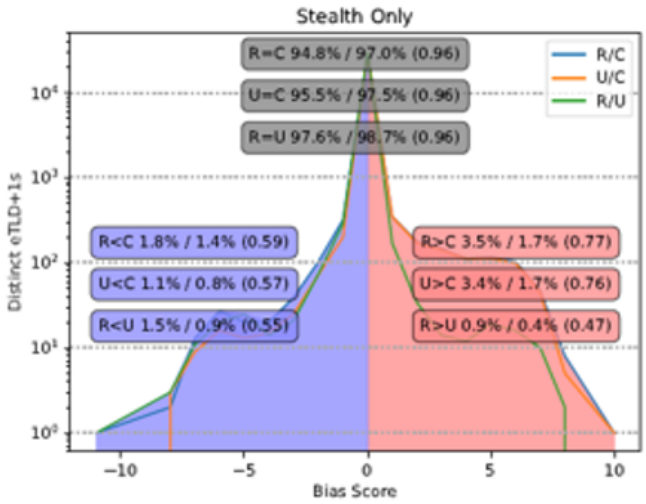

Figure 8.2: Graphical visualisation of quantification using a population pyramid, Source: [JSS+21]

figurations. On the vertical axis, we see the number of websites for which there is a particular bias. The graph further contains three sets of curves that represent the comparison between the three combinations of VPs (Residential vs Cloud-based (R/C), University vs Cloud-based (U/C), and Residential vs University (R/U)). The boxes inside each graph provide additional detailed bias information. For example, between the Residential vs Cloud-based, there is a pro-residential bias of 3,5 %, which is observed on 1,7 % of all domains.

The overall conclusion that we can draw from this graph is that there is a strong pro-residential bias compared to the cloud VP. In aggregate, these biases account for significant volumes of traffic.

Evaluation: The choice for the logarithmic "bias score" that the study by Jeuckstock et al. use has a sound basis. It makes it easier to compare two variables and its descriptive power makes it easier to draw conclusions. It also avoids extreme outliers.

From the coloured areas, we can get a quick general impression of the distribution of the number of websites for a certain bias. It also allows us to judge the symmetricalness or a specific bias to one side.

A graph with two sides. That is optimal to represent those impacting factors from which there are also two sides (or types), which are the yellow ones in Figure 8.1. For example, you could include or exclude web scraping history in your results. On the contrary, if you want to compare scraping results from many different locations or vantage points, you will quickly end up with many charts, all combining two variables, or you would clog up one chart with many curves. If you combine many curves in one chart, then it is not very clear what data you compare.

WEB BOT DETECTION - FINGERPRINTING

To study web bot detection, we can compare the web scraping results of two scrapes, one from a standard web bot and one from a stealthy web bot (resembles more to a human). The study by Zeber et al. [ZBO+20] compared web scraping results from a web bot against real human web browsing. For the discussion, we will limit ourselves to the metric of third-

party domains.

The authors aggregated the web scraping results to compare the differences to obtain an average number of third-party domains for each visited domain. Figure 8.3 shows the distribution of these average values across unique domains in a graph type, called box plots.

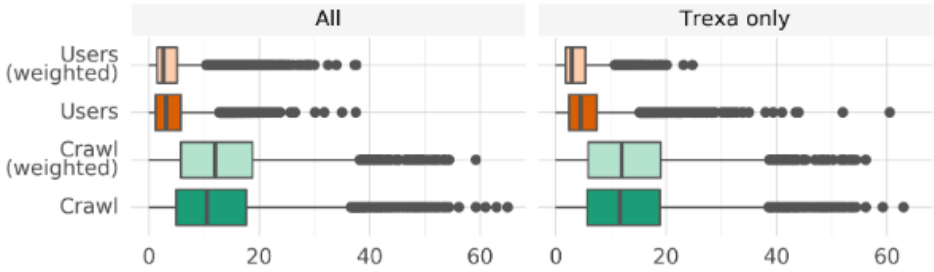

Figure 8.3: Use of box plot graphs to compare distributions over visited domains, Source: [ZBO+20]

The figure shows left and right two different sections of domains (all domains and only TREXA10KU). There exist a weighted version of the results in each subset, where all the domains are weighted equally. In the other version (weighted), the authors re-sampled all the domains according to the number of visit counts.

Looking at the box sizes, we can quickly spot significant differences in the number of visited third-party domains. In particular, the median (line in the box) in the TREXA10KU subset for the crawl site visits is 11,6, almost threefold for that of human visits (4,5 third parties). For the weighted version of human visits, this number drops to 2,9. Zeber et al. conclude that this could be because users may see different portions of websites.

Evaluation: Box plots offer many advantages for this kind of research. They make it easy to get an overview and compare different related data sets. Additionally, its use comes with five statistical numbers, three quartiles (which includes a median), a minimum, and a maximum. In the example, in Figure 8.3, we also appreciate the advantage of the horizontal scale that allows us to relate to the real metric.

Despite these advantages, the graph's simplicity, we are limited to observe the data's distribution details. We aggregate all data into the box plot and miss out on the details that we could observe in Figure 8.2. Besides that, the level of detail depends heavily on the size of the picture. For example, look at the small difference between the two upper boxes on the right side of Figure 8.3. We need to compare the medians to see a difference.

In Figure 8.3 we see four boxes that highlight the advantage of box plots over population pyramids, where we would need six graphs to compare four variables with each other. We thus can apply this type of graph to nearly all the location-related impacting factors (green). In our discussion on the results, we did a one by one comparison. So it is possible to use the graph for the yellow-colour impacting factors, but that is not where its strength lies.

Dynamic web - Content updates

Besides the difference between web bots and humans, one of the most noticeable results of the study by Zeber et al. [ZBO+20] is the influence of time on the metrics. Web content changes over time, and in our taxonomy, we ascribe the cause for the impact of time to the dynamic web, specifically to the content updates.

The authors measured the effects of time by performing thirty-nine web scrapes over a period of fifty-four days. They quantified the impact of time by comparing the scraping results against a baseline in a box plot graph. In Figure 8.4 Zeber et al. divided this period into four discrete parts. In each graph (for each metric), for each part, they plotted the distribution of the mean Jaccard similarity in a box. The Jaccard similarity calculates the difference between the baseline and the web scraping results.

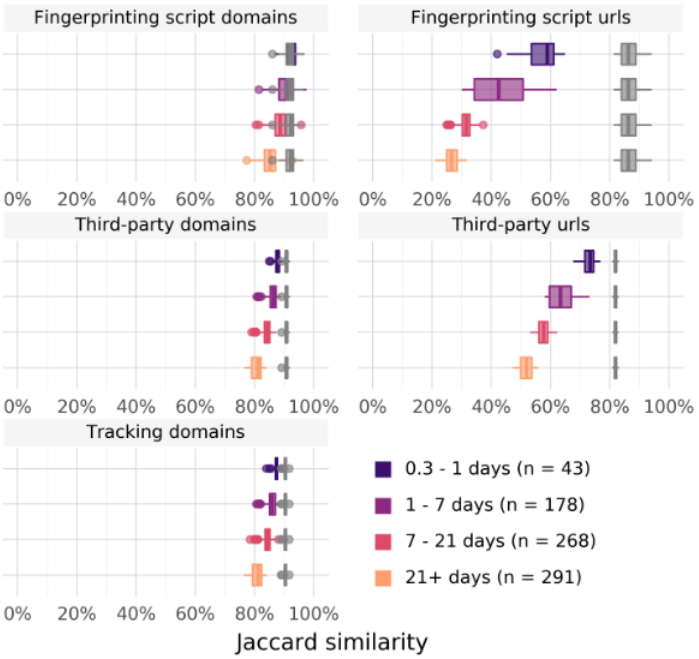

Figure 8.4: Box plots comparing the mean Jaccard similarity against a baseline for the time metric, Source: [ZBO+20]

Overall, but especially in the two graphs on the right side, we see that the distributions shift to the left as time increases. Such a shift means that the metrics deviate stronger from the baseline as time passes. The baseline range is above eighty for the fingerprinting script URLs metric, while the median shifts from about sixty on the first day to thirty after twenty-one days.

Evaluation: We already reviewed box plot graphs in the previous subsection, but this example shows an interesting variation. First of all, measuring change over time, but also comparing box plots against a baseline. Another positive point in this example is that the use of Jaccard similarities introduces a standard way to present the results, which improves readability and understanding. Nevertheless, if the box overlaps with the baseline, then it becomes unreadable, as in the top-left graph in Figure 8.4. Furthermore, you need a baseline for comparison with your data.

Here, Zeber et al. quantify content updates over time. Content updates have a yellow colour in Figure 8.1, which means that we would compare two web scraping results. And this is also what we see in our example, be it that it repeats this over four time periods. You could do this with all yellow-coloured impacting factors, but also for the green ones (for a single location). Also, for the location-related impacting factors (green), you could compare the results of multiple locations against a baseline of one location.

8.2. CONCLUDING RECOMMENDATION

After our short evaluation of four quantification methods, we can see that each method has distinct advantages. For example, tabular results have the advantage of showing the numbers in great detail. However, we want to be able to get a quick overview of all results, and we might not have enough data to calculate a baseline.

So, there remain the two methods that we have seen in Figure 8.2 and 8.3. Both are great methods that allow us to get an overview at a glance to draw meaningful conclusions. But then again, both have their limitations when we judge them in the light of general applicability to our impacting factors. The population pyramid shows a breakdown of the data but is most suited for one on one comparisons. The box plots allow us to compare multiple data sets (so also two, although not optimal), but because of its simplicity, it lacks the detailed distribution information that we see in the population pyramid. Above that, if we have only two variables to compare, then we cannot take advantage of the full power of box plots. Or, in other words, box plots are not really suited for a one on one comparison.

These shortcomings forced us to look for an alternative. We set the requirements for this alternative that the quantification method is generically applicable to most of the impacting factors. We said that we would focus on the yellow and green impacting factors. Next, we want to be able to make a quick comparison, and, furthermore, we want to get a more detailed grasp of the distribution of the data.

For our requirements, we found the perfect match: violin plots. Violin plots are closely connected to box plots, as you can see in Figure 8.5. Violin plots use density curves to plot the data's distribution. It allows us to compare the distribution of multiple groups. We can plot additional overlay information, such as the box of a box plot or only some of its features, such as the three quartile lines (which reduce the visual noise).

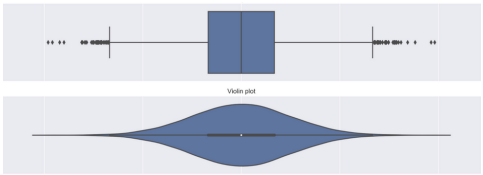

Figure 8.5: Comparison of a box plot and a violin plot, Source: https://towardsdatascience.com/violin-plots-explained-fb1d115e023d

Using violin plots, we can now specify, at a high level, a generic quantification technique to measure the impact of the impacting factors and mitigations.

- We start with the assumption that the input data consists of the web scraping results of a set of parallel (and synchronised) web scrapes to rule out any incidental data resulting from network errors etc. Jueckstock et al. [JSS+21] use three web scrapes, and Krumnow et al. [KJK22] eight. From these, we take the mean of the collected data to keep that data that is consistent across all web scrapes.
- The mean data becomes the input to our graph, which will be a violin plot type graph overlayed with the three quartile lines of a box plot.
- We can draw general conclusions by comparing the distributions in the graph. We can furthermore draw exact conclusions by calculating the percentile difference between the numeric means (quartile two) of two violins.

9

Discussion

Our research's main aim was to improve web measurements. We tried to do this by offering future researchers in the same field as our scope a complete set of impacting factors and mitigations. They, in their turn, would be able to choose the most relevant to their study, to improve their web measurements.

We searched the relevant literature within our scope for confounding variables. While we saw three to four confounding factors in each study, we were surprised to find a total of thirty-one impacting factors. The consulted literature backs about two-thirds; we found the others by brainstorming with subject matter experts. While thirty-one impacting factors cover whole the spectrum of a generic scraping model, we must note that we were seriously restricted by the duration of the project and the topics within our scope. We expect that future work can reveal more impacting factors.

For each of the impacting factors, we proposed a mitigation technique. We found the corresponding mitigation techniques in a similar way as we found the impacting factors. Overall, we found effective mitigations. However, in just under half of the cases, the mitigations tend to solve the problems only partially. Just for two instances, the mitigations were not reliable. We discovered that we could not prevent error-related factors. Any of the related mitigations thus involve the discovery of those errors to be able to take appropriate actions. We, additionally, also find similarities in the type of mitigations for location-related impacting factors. Here, we mainly rely on the use of a VPN Proxy.

We started our further analysis of the impacting factors by categorising them into a taxonomy. We derived its highest level categories (client side, server side, network) from the three major areas of our scraping model. Lower level categories came about grouping our impacting factors. We were also able to derive further knowledge from this taxonomy. For example, we see that the client-side factors are the most accessible and easiest to mitigate. Although strictly separate issues, we see error categories in each of the major categories. Also, the impact of location seems to apply to both outer ends of the scraping model, which are the client side and the server side. However, we base the discovery of this location on the IP address, which again is mostly a networking factor. That is why we categorised all the related impacting factors under the network branch. The mitigation for all these factors comprises the use of a VPN Proxy. We extended this analysis of the relation between the categories and mitigations. For a few categories, we found a relation between their impacting factors and the mitigations, for example, with the location and dynamic web categories.

These findings require us, however, to abstract away from the solutions to find a common theme. We finally conclude that the taxonomy also provides the basis for searching for unknown factors (which we did not discover yet). We discussed a few techniques that we can use to find such unknown impacting factors.

Furthermore, we concluded that if the unknown factors fall within such categories, as we described in our previous point, we can also find mitigations for the unknown factors. A final limitation that we want to note is that a taxonomy is a living document that is updated over time by different users and after the lessons learned by its use. We, on the other hand, were limited to its development with the thesis project.

We finally took our analysis one step further to discuss how we could quantify the impact of our impacting factors and mitigations. Here we concluded that for most impacting factors and mitigations, the quantification of their impact requires us to compare web scrapes. But this is what most involved studies already do, so they provide ready to use methods.

In contrast, for error-related factors that are not supposed to occur, we can quantify by counting the number of occurrences. We witnessed a general trend in the studies that data from web scrapes where errors occur, are omitted from the final results. Additionally, we generally do not want permanent countermeasures, such as blacklisting, to occur. So, we question whether we really want to quantify these types of factors.

For the impacting factors, for which we can compare web scrapes, we were able to find a quantification technique that is mainly based on the analysis of violin plot graphs. These kinds of graphs combine the advantages of box plots and population diagrams. Nevertheless, violin plots are a bit harder to interpret than regular box plots, and they are also visually noisier.

10

CONCLUSIONS AND FUTURE WORK

Our research studied the factors that impact web measurements used in privacy and user tracking-related studies.

We performed and extensive structured literature study that resulted in the identification of a total of thirty-one impacting factors and corresponding mitigations. The development of a scraping model, based on the literature, allowed us to set the scope for and scrutinise the relevant impacting factors. We later processed the scraping model and impacting factors into a taxonomy. Further analysis of the taxonomy revealed relations between categories and the mitigations of the impacting factors within them. These relations allowed us to developed some techniques to find mitigations for still unknown impacting factors. We finally developed a generic quantification technique based on violin plots that would allow us to measure the impact of most impacting factors and mitigations.

Our conclusions drive future researchers to expand on their limited view of confounding variables to create a complete picture of what factors can affect their study. With this, we aim to allow researchers to improve their web measurements by taking into account and mitigating whatever impacting factors could poison their results.

FUTURE WORK

This study is far from finished. Future work should expand on the studied topics and refine all identified impacting factors and mitigations. Putting the results to work and processing lessons learned would certainly enhance the quality of our work. A taxonomy is also never truly finished. Extending it and updating it is a requirement so that it truly can help other researchers improve their web measurements. Finally, we proposed a starting point for a total quantification technique to measure the impact of the impacting factors and mitigations. Holding our results in the light of the quantification methods of numerous other studies can further prove its applicability. On the other hand, if it does not generally apply, we might discover some better alternatives.

11

REFLECTION

I started the graduation project with a detailed plan from my research proposal and a lot of motivation to follow it. After the first phase, the study of the theory and starting to write took so long, that I could not spent any time on the practical (coding) aspects of my research. In the following phase, I missed the basis to carry out the rest of the plan on time. Two months in the second phase, I dropped the whole planning completely. I really did not expect that I would need so much time to study the literature.

From the original research questions, from the research preparation module, I actually based whole this thesis on the first one. Now, at the end of the graduation project, I am really amazed about how much effort this took. When we decided to only focus on the confounding factors, I did not really have a problem with that, but I found it a true setback that all the work that I put in my research proposal was now useless.

I am further on satisfied that I could put on through to finish the graduation project until the end. I find the results quite satisfying, although I did not produce any code (which was a requirement for the module). I am not really sure how that will affect the final score.

A

SUMMARY OF MITIGATIONS

In the following table, we present a complete overview of all the impacting factors that we identified. We ordered them according to the branches and categories in our taxonomy. We furthermore present our preferred method of mitigating these impacting factors. Where it applies, we give an example of the study that inspired the mitigation. We finally indicate whether the mitigation will truly mitigate the impacting factors with three categories, a checkmark (fully), partially, and not quite.

Confounding factors	Mitigation control and source	Resolvable?
Factors related to the scraping study's design		
Configuration parameters		
Bias in website selection	Random selection	✓
Normalised URL format	Use only host name (incl. path) [ZBO+20]	✓
Amount of scraping traffic	Diversify IP addresses (VPN Proxy) and limited parallel scrapes [VNBJ14, ZBO+20, JSS+21]	✓
Equivalence of web scrapes	Explicitly define "equivalence" for the study	✓
Factors related to the client side		
Web page dynamics		
Cookie Dialogues	Assist in manual consent or use browser plugin	✓

Life updates	Synchronised and parallel web scrapes	✓
Infinite scroll	Ignore to avoid web bot detection	/
Errors		
Browser errors and crash	OpenWPM standard recovery [EN16]	✓
Scraping framework crash	Error monitoring and logging	Partially
Scraping database crash	Error monitoring and logging	Partially
Whole computer system slow-down or crash	Error monitoring and logging	Partially
Attacks		
Attacks on data collection mechanism	Use detection mechanisms [KJK22]	Partially
Web bot detection		
Browser Fingerprinting	Use stealthy plugin [JSS+21, KJK22]	partially
Behavioural based web bot detection	Do not perform any page interactions, use OpenWPM extensions [KJK22]	Partially
History		
Scraping history	Use OpenWPM stateless crawl configuration [EN16]	✓
Client-side caching	Clear cache between scraping runs, one web bot per cache	✓
Network-related factors		
Errors		
Network errors	Error monitoring and logging, longer timeouts against congestion [RL18, JSS+21]	partially

Request/ response modification		
ISP advertising	Use ad blocker, VPN proxy	✓
Network caching	Do not use enterprise networks	/
Middleboxes	Use residential network as baseline	✓
Client-location		
Vantage point	Use residential IP address or rotating proxies to residential address pool	✓
IP address blacklisting or blocking	VPN proxy for most detectable bots and check IP reputation [JSS+21, ADZ+20]	Partially
DNS resolution	Use VPN Proxy, custom DNS resolver [JSS+21]	Partially
Geo-targeting	use of VPN proxy that lies in the same geopolitical region [JSS+21, EN16, ZBO+20, ADZ+20]	✓
Server-location		
Content Delivery Networks/ Datacentre	Keep web bot distribution within same geographical area and Synchronise scrapes	Partially
Local Privacy Regulations and legislation	Keep web bot distribution within same geographical area, VPN Proxy	✓
Factors related to the server side		
Dynamic web		
Changes in content or layouts and updates	Synchronise scrapes [FBLS20, JSS+21, MGEL12, ADZ+20, KJK22]	✓
Individualisation	Generic scraping [JSS+21]	Partially

A/B testing	Parallel scraping [HTWH18, JSS+21, ZBO+20, FBLS20]	✓
Cloaking	Compare web bot scrapes against human browsing [WD05], use less detectable web bot [KJK22]	Partially
Errors		
Server non-responsive	Monitor *5xx* HTTP status codes	Partially

Table A.1: Identified confounding variables and possible mitigations

BIBLIOGRAPHY

[ADZ+20] Syed Suleman Ahmad, Muhammad Daniyal Dar, Muhammad Fareed Zaffar, Narseo Vallina-Rodriguez, and Rishab Nithyanand. Apophanies or epiphanies? how crawlers impact our understanding of the web. In Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen, editors, *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pages 271–280. ACM / IW3C2, 2020.

[AEE+14] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juárez, Arvind Narayanan, and Claudia Díaz. The web never forgets: Persistent tracking mechanisms in the wild. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 674–689. ACM, 2014.

[Ari21] Dror Arie. Content delivery network explained, 2021.

[BDGP19] Desamparados Blazquez, Josep Domenech, José A. Gil, and Ana Pont. Monitoring e-commerce adoption from online data. *Knowl. Inf. Syst.*, 60(1):227–245, 2019.

[CB02] Brian E. Carpenter and Scott W. Brim. Middleboxes: Taxonomy and issues. *RFC*, 3234:1–27, 2002.

[CDT08] CDT. Online behavioral advertising: Discussing the isp-ad network model, 2008.

[CGW18] Zi Chu, Steven Gianvecchio, and Haining Wang. Bot or human? A behavior-based online bot detection system. In Pierangela Samarati, Indrajit Ray, and Indrakshi Ray, editors, *From Database to Cyber Security - Essays Dedicated to Sushil Jajodia on the Occasion of His 70th Birthday*, volume 11170 of *Lecture Notes in Computer Science*, pages 432–449. Springer, 2018.

[Con70] Kate Conger. Uber's massive scraping program collected data about competitors around the world. *gizmodo*, 2070.

[Dob10] Jaromir Dobias. Privacy effects of web bugs amplified by web 2.0. In Simone Fischer-Hübner, Penny Duquenoy, Marit Hansen, Ronald Leenes, and Ge Zhang, editors, *Privacy and Identity Management for Life - 6th IFIP WG 9.2, 9.6/11.7, 11.4, 11.6/PrimeLife International Summer School, Helsingborg, Sweden, August 2-6, 2010, Revised Selected Papers*, volume 352 of *IFIP Advances in Information and Communication Technology*, pages 244–257. Springer, 2010.

[Eck10] Peter Eckersley. How unique is your web browser? In Mikhail J. Atallah and Nicholas J. Hopper, editors, *Privacy Enhancing Technologies, 10th International Symposium, PETS 2010, Berlin, Germany, July 21-23, 2010. Proceedings*, volume 6205 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2010.

[EN16] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1388–1401. ACM, 2016.

[FBLS20] Imane Fouad, Nataliia Bielova, Arnaud Legout, and Natasa Sarafijanovic-Djukic. Missed by filter lists: Detecting unknown third-party trackers with invisible pixels. *Proc. Priv. Enhancing Technol.*, 2020(2):499–518, 2020.

[FMSB15] Nathaniel Fruchter, Hsin Miao, Scott Stevenson, and Rebecca Balebako. Variations in tracking in relation to geographic location. *CoRR*, abs/1506.04103, 2015.

[Hed10] Heather Hedden. *The accidental taxonomist*. Information Today, New Jersey, 2010.

[Hed20] Heather Hedden. Classification systems vs. taxonomies, 2020.

[HTWH18] Thomas Hupperich, Dennis Tatang, Nicolai Wilkop, and Thorsten Holz. An empirical study on online price differentiation. In Ziming Zhao, Gail-Joon Ahn, Ram Krishnan, and Gabriel Ghinita, editors, *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy, CODASPY 2018, Tempe, AZ, USA, March 19-21, 2018*, pages 76–83. ACM, 2018.

[ITK+16] Luca Invernizzi, Kurt Thomas, Alexandros Kapravelos, Oxana Comanescu, Jean Michel Picod, and Elie Bursztein. Cloak of visibility: Detecting when machines browse a different web. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 743–758. IEEE Computer Society, 2016.

[JKV19] Hugo Jonker, Benjamin Krumnow, and Gabry Vlot. Fingerprint surface-based detection of web bot detectors. In Kazue Sako, Steve A. Schneider, and Peter Y. A. Ryan, editors, *Computer Security - ESORICS 2019 - 24th European Symposium on Research in Computer Security, Luxembourg, September 23-27, 2019, Proceedings, Part II*, volume 11736 of *Lecture Notes in Computer Science*, pages 586–605. Springer, 2019.

[JSS+21] Jordan Jueckstock, Shaown Sarker, Peter SnyderΔ, Aidan Beggs, Panagiotis Papadopoulos, Matteo Varvello, Ben LivshitsΔ, and Alexandros Kapravelos. Towards realistic and reproducible web crawl measurements. *ACM*, 2021.

[Kha17] Sarah Khan. An introduction to taxonomies, 2017.

[KJK22] Benjamin Krumnow, Hugo Jonker, and Stefan Karsch. All browsers are equal... or is openwpm less equal than others? In *currently under review*, 2022.

[KJS20] Vlad Krotov, Leigh Johnson, and Leiser Silva. Tutorial: Legality and ethics of web scraping. *Commun. Assoc. Inf. Syst.*, 47:22, 2020.

[LSKR16] Adam Lerner, Anna Kornfeld Simpson, Tadayoshi Kohno, and Franziska Roesner. Internet jones and the raiders of the lost trackers: An archaeological study of web tracking from 1996 to 2016. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*. USENIX Association, 2016.

[MGEL12] Jakub Mikians, László Gyarmati, Vijay Erramilli, and Nikolaos Laoutaris. Detecting price and search discrimination on the internet. In Srikanth Kandula, Jitendra Padhye, Emin Gün Sirer, and Ramesh Govindan, editors, *11th ACM Workshop on Hot Topics in Networks, HotNets-XI, Redmond, WA, USA - October 29 - 30, 2012*, pages 79–84. ACM, 2012.

[Mit18] Ryan Mitchell. *Web scraping with Python: Collecting more data from the modern web*. O'Reilly Media, Inc., 2018.

[MM12] Jonathan R. Mayer and John C. Mitchell. Third-party web tracking: Policy and technology. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pages 413–427. IEEE Computer Society, 2012.

[MWRK10] Gabriel Maciá-Fernández, Yong Wang, Rafael Rodríguez-Gómez, and Aleksandar Kuzmanovic. Isp-enabled behavioral ad targeting without deep packet inspection. In *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA*, pages 1469–1477. IEEE, 2010.

[PKM18] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. Cookie synchronization: Everything you always wanted to know but were afraid to ask. *CoRR*, abs/1805.10505, 2018.

[PSF16] Kien Pham, Aécio S. R. Santos, and Juliana Freire. Understanding website behavior based on user agent. In Raffaele Perego, Fabrizio Sebastiani, Javed A. Aslam, Ian Ruthven, and Justin Zobel, editors, *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 1053–1056. ACM, 2016.

[RL18] Jukka Ruohonen and Ville Leppänen. Invisible pixels are dead, long live invisible pixels! In David Lie, Mohammad Mannan, and Aaron Johnson, editors, *Proceedings of the 2018 Workshop on Privacy in the Electronic Society, WPES@CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 28–32. ACM, 2018.

[TJM15] Christof Ferreira Torres, Hugo L. Jonker, and Sjouke Mauw. Fp-block: Usable web privacy by controlling browser fingerprinting. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part II*, volume 9327 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2015.

[Vlo18] Gabry Vlot. Automated data extraction; what you see might not be what you get. Master's thesis, Open Universiteit, 2018.

[VNBJ14] Thomas Vissers, Nick Nikiforakis, Nataliia Bielova, and Wouter Joosen. Crying wolf? on the price discrimination of online airline tickets. In *7th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2014)*, 2014.

[WD05] Baoning Wu and Brian D. Davison. Cloaking and redirection: A preliminary study. In *AIRWeb 2005, First International Workshop on Adversarial Information Retrieval on the Web, co-located with the WWW conference, Chiba, Japan, May 2005*, pages 7–16, 2005.

[WSV11] David Y. Wang, Stefan Savage, and Geoffrey M. Voelker. Cloak and dagger: dynamics of web search cloaking. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011*, pages 477–490. ACM, 2011.

[YA08] Jeff Yan and Ahmad Salah El Ahmad. Usability of captchas or usability issues in CAPTCHA design. In Lorrie Faith Cranor, editor, *Proceedings of the 4th Symposium on Usable Privacy and Security, SOUPS 2008, Pittsburgh, Pennsylvania, USA, July 23-25, 2008*, ACM International Conference Proceeding Series, pages 44–52. ACM, 2008.

[ZBO+20] David Zeber, Sarah Bird, Camila Oliveira, Walter Rudametkin, Ilana Segall, Fredrik Wollsén, and Martin Lopatka. The representativeness of automated web crawls as a surrogate for human browsing. In Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen, editors, *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pages 167–178. ACM / IW3C2, 2020.