Karlsruhe Institute of Technology

Better Web Measurements: Enhancing Security and Privacy Research

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Nurullah Demir

geb. in Midyat

Tag der mündlichen Prüfung: 20. Oktober 2023
Erster Gutachter: Prof. Dr. Christian Wressnegger
Zweiter Gutachter: Prof. Dr. Martin Johns

“Oğlum, sen git, yeter ki oku, ben gerekirse ceketimi satarım...”

— Mehmet Nezir Demir

Teşekkür ederim baba.

Acknowledgments

Achieving this long dreamed of goal would not have been possible alone. First and foremost, I would like to express my heartfelt gratitude to my esteemed parents' unfailing support. The entire process would have been far more challenging without my father's innovative spirit and my mother's heartfelt whispers.

I owe a huge debt of gratitude to my esteemed advisor, Prof. Dr. Norbert Pohlmann, for his support since my early education years, for opening the doors to this doctoral education, and for laying the groundwork. I benefited immensely from the environment he provided, reflecting his liberal views. A special thanks to Prof. Dr. Christian Wressnegger for the invaluable time he dedicated to me, the discussions, and his constant fruitful assistance. Working with him has been a unique and enriching experience. I also appreciate Prof. Dr. Martin Johns for his valuable time and support in the final stages of this study.

I would like to thank Dr. Tobias Urban for our numerous productive discussions since the first day of my doctoral education. I learned a lot from him and appreciate his support and collaboration. I would also like to thank my co-authors, Prof. Dr. Thorsten Holz, Prof. Dr. Matteo Große-Kampmann, and Jan Hörnermann. Additionally, I am grateful to my dear friends Dominik Adler, Ines Hamdi, İsmail Çolak, Yusuf Seyhan, and Onur Güler for their contributions, companionship, and collaboration during this period.

Moreover, to my wonderful wife, Meryem: since the first day of my educational years, you have not only been patient with me but also understanding, constantly supportive, and always by my side. I cannot imagine any of this without you. I am deeply thankful for everything. Also, to my son, Tahir: you boosted my motivation on this journey and made everything more fun and colorful. Thank you for that. Everything is much more unique and beautiful with you all.

Abstract

The World Wide Web has revolutionized our access to and sharing of information, our connections with each other, and our ways of doing business in the modern age of technology. As the Web is gaining importance and collecting more sensitive data, it becomes increasingly crucial to Web users to understand its impact on security and privacy. In this context, large-scale Web measurements are powerful tools for exploring the Web environment. Large-scale measurements provide valuable insights into the strengths and weaknesses of the Web. Such measurements, however, represent a major challenge due to the inherent complexity of the Web's architecture, varying local regulations, and the diversity of measurement environments. These factors can quickly bias results and render study findings inconsistent.

This thesis is aimed at comprehending factors leading to variances and pinpointing feasible methods to conduct more robust and thus verifiable security- and privacy-related Web measurements. We employ *Metascience* methodologies to examine how the robustness of studies can be enhanced, thereby yielding more reliable insights for researchers, developers, and end users. To this end, we conduct various large-scale Web measurements while crawling 17 million webpages across the global Web.

We first tackle the aspect of *robustness* through two comprehensive large-scale measurement studies. These studies were based on running 24 different measurements across 4.5 million pages and scrutinizing the loading dependencies of 1.7 million webpages with five profiles. We highlight how minimal design choices in a Web measurement study can significantly influence the results and identify resource characteristics that amplify these variances. We finally provide guidelines for enhancing robustness of such studies.

Expanding on the robustness findings, we launched an 18-month Web *security* measurement that covered over 5.6 million websites and 246 distinct software components. This case study underlines that 95% of the evaluated websites utilize at least one vulnerable software component. Furthermore, it validated that conducting measurements on a static dataset is the only reliable approach to completely reproducing a Web measurement.

To tackle the *privacy* aspect, we carried out another case study examining 2.6 million webpages across 27 profiles. Our analysis compared various "cookie banner interaction" browser extensions and their effectiveness in communicating user preferences regarding cookies. Our results revealed significantly different effects, even between extensions offering the same functions, and identified side effects that may compromise user privacy.

Zusammenfassung

Das World Wide Web hat den Zugang zu und Austausch von Informationen im Technologiezeitalter neu definiert. Mit wachsender Relevanz und Datensammlung wird das Verständnis für Sicherheit und Datenschutz im Web immer bedeutsamer. In diesem Zusammenhang sind groß angelegte Webmessungen mächtige Methoden, um die Webumgebung zu erkunden. Sie bieten wertvolle Einblicke in die Stärken und Schwächen des Webs. Solche Messungen stellen jedoch aufgrund der inhärenten Komplexität der Web-Architektur, variierender lokaler Regulierungen und der Vielfalt der Techniken eine große Herausforderung dar. Diese Faktoren können Ergebnisse schnell verzerren und zu inkonsistenten Studienergebnissen führen, was eine sorgfältige Durchführung erfordert.

Diese Arbeit zielt darauf ab, die Faktoren zu verstehen, die zu Abweichungen führen, und praktikable Methoden zu identifizieren, um robustere und somit überprüfbare sicherheits- und datenschutzbezogene Webmessungen durchzuführen. Wir verwenden *Metascience*-Methoden, um zu untersuchen, wie die Robustheit von Studien erhöht werden kann, und liefern somit zuverlässigere Erkenntnisse für Forscher, Entwickler und Endbenutzer. Zu diesem Zweck führen wir verschiedene groß angelegte Webmessungen durch.

Zunächst gehen wir auf den Aspekt der *Robustheit* durch zwei Messstudien ein. Sie basieren auf 24 verschiedenen Messungen auf 4,5 Millionen Seiten und der Überprüfung der Ressourceabhängigkeiten von 1,7 Millionen Webseiten mit fünf Profilen. Wir zeigen, wie minimale Designentscheidungen in einer Studie das Ergebnis erheblich beeinflussen können und identifizieren Ressourcenmerkmale, die diese Varianzen verstärken. Schließlich geben wir Empfehlungen für die Erhöhung der Robustheit solcher Studien.

Basierend auf den Erkenntnissen zur Robustheit führten wir eine 18-monatige *Sicherheitsmessung* durch, die über 5,6 Millionen Webseiten und 246 unterschiedliche Software abdeckte. Diese Fallstudie zeigt, dass 95% der Webseiten mindestens eine verwundbare Softwarekomponente einsetzen. Wir verdeutlichen, dass Messungen an statischen Datensätzen der einzige zuverlässige Ansatz zur kompletten Sicherstellung der Reproduzierbarkeit ist.

Um den *Datenschutz*-Aspekt anzugehen, führten wir eine weitere Fallstudie durch, bei der 2,6 Millionen Webseiten mit 27 Profilen untersucht wurden. Unsere Analyse verglich verschiedene "Cookie-Banner-Interaktion"-Browsererweiterungen und deren Wirksamkeit bei der Kommunikation von Nutzerpräferenzen. Unsere Ergebnisse zeigen signifikant unterschiedliche Effekte, und identifizieren Nebenwirkungen, die die Privatsphäre des Benutzers gefährden können.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The World Wide Web (WWW, or the Web) has brought a paradigm shift in communication and data exchange, which profoundly impacted various aspects of modern life. Today, the Web is a platform for various purposes, from personal communication to entertainment via streaming services to facilitating business operations. As the Web continues to evolve, the number of populations with Web access is also increasing [157], with the substantial benefits offered by the Web playing a significant role in our daily lives.

Since this enrichment increases the volume of sensitive information processed via the Web and given the Web's pivotal role in contemporary society, it is imperative to comprehend its multifaceted components and their impact on security and privacy. This includes identifying the prevailing challenges, formulating responsive actions, and implementing optimization strategies to enhance its safety and privacy. An effective approach to achieving this is to conduct large-scale Web measurements using Web crawlers (e.g., *Wget* [55], *cURL* [29], *Selenium* [147]), which enable data collection on webpages. These measurements provide valuable insights based on real-world data and help us understand emerging trends and challenges in the online ecosystem.

Various stakeholders (e.g., developers, end users) utilize these insights to guide their actions, which underlines the global impact of such measurements. However, carrying out these measurements is inherently complex. Numerous factors have a direct impact on the results (e.g., utilized tools, local regulations) [162, 4]. When overlooked, these factors can introduce biases to the findings, which may adversely affect the stakeholders (e.g., developers, end users). Understanding the factors that lead to variances in Web measurements is essential for obtaining reliable results. Such factors include the natural complex architecture of the Web [113], local legal regulations [47, 150], varied features and techniques of tools [44, 58, 173, 126, 137], and different measurement strategies [4]. Thus, the dynamic and complex architecture of the Web, along with various challenges such as controlling confounding factors, ensuring reproducibility, and maintaining transparency, pose considerable difficulties in different stages of a Web measurement study [22].

In this context, it is crucial to gain a deep understanding of the variance in Web measurements, investigate contributing factors, and explore ways to enhance the "robustness" of Web measurements and minimize bias in the results. In this thesis, we therefore explore methods to improve the reliability of Web measurements in the realm of security and privacy, aiming to provide more reliable insights to the key stakeholders of the Web.

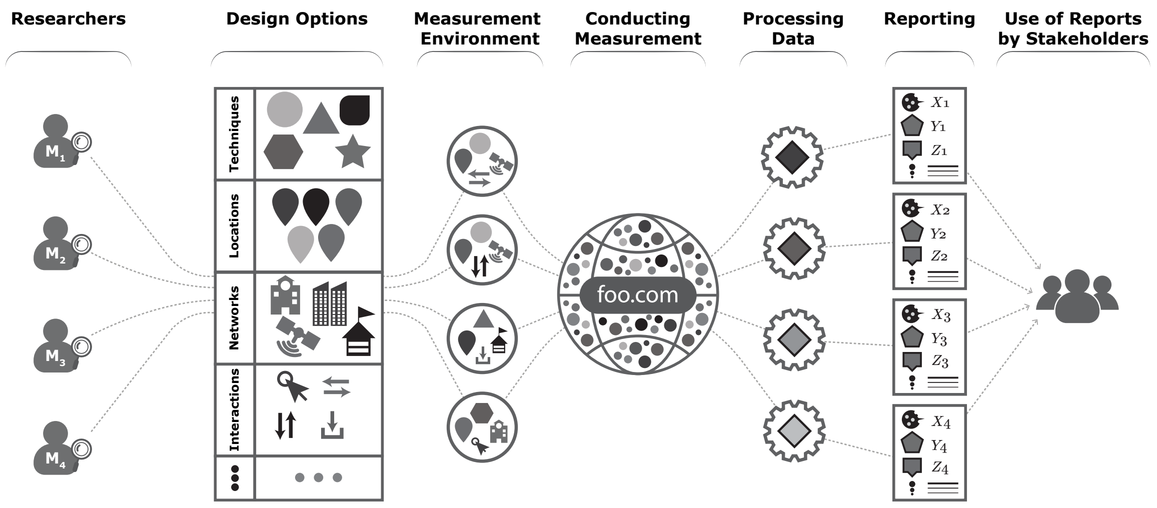

Figure 1.1.: Current challenges in life cycle of Web measurement studies.

1.1. Intricacies of the Web and Their Impact on Web Measurements

We now explore factors that lead to variances in Web measurements under the influence of the Web's inherent structure and measurement design choices. A visual representation of the challenges in life cycle of a Web measurement study can be seen in Figure 1.1.

The way data travels across the Web is influenced by its infrastructure, which includes components such as undersea cables and satellite networks. This can introduce variances in latency, such as ping times, connection speeds, and data loss, depending on the chosen transmission path [51]. Cloud platforms with their diverse architectures and technologies can lead to differences in data access times, processing speeds, and data consistency [46]. *Content Delivery Networks* (CDNs) employ various algorithms and strategies for content distribution. This can result in variances in content delivery times, cache refresh rates, and regional content optimization [141]. DNS servers, depending on their hierarchy and location, might exhibit different resolution times, caching strategies, and error rates [3]. Web servers, contingent on their hardware and software configurations, can differ in processing speeds, response strategies, and resource utilization [50, 27]. The multitude of web browsers interprets and processes Web content in their unique ways. This can lead to variances in content rendering and user-web interactions [136]. The dynamics of websites means that sites often adjust their content based on user profiles, behavioral data, or geographical locations. This dynamics can introduce variances in content display and website loading times [82]. Regulatory mandates set by authorities and organizations can cause regional differences in Web accessibility, content filtering, and security protocols [87].

Apart from these components, several other factors contribute to variances in Web measurements. We now focus on aspects primarily influenced by the design choices of researchers when conducting such measurements. The data source utilized, such as top lists from sources like *Tranco* or *Alexa*, can inherently introduce biases, with some sources potentially offering a more comprehensive or skewed perspective [100]. The tools and frameworks employed, ranging from *Selenium*, *OpenWPM* to *Puppeteer*, can significantly impact on the type and accuracy of data collected [87]. The choice of network, be it a university network, VPN, government infrastructure, or a private network, can introduce variances due to speed, reliability, and potential filtering or throttling differences [164, 103]. The specific metrics chosen for measurement, such as tracking requests or HTTP security headers, can also lead to different outcomes, as some metrics might be more susceptible to external influences than others [64]. The measurement strategy, which might involve interactions with the target, and the measurement design, like designing a specific measurement pipeline, play crucial roles in determining the robustness and consistency of results [4]. Other factors, such as relying on static data sources like the *HTTPArchive.org, Archive.org* or dynamically measuring websites, may also influence the measurements [64].

Each factor can introduce variances in Web measurements, which highlights the need for thorough and context-sensitive analysis. Consequently, the demand for robust Web measurements that account for these and other considerations is evident. In the subsequent sections, we delve into robustness within a scientific context, exploring potential avenues to achieve more robust Web measurements.

1.2. Understanding Robustness

Above all, it is necessary to define *robustness* within the context of scientific studies. Robustness is an element of scientific practices that remains invariant under partly independent multiple determinations and refers to the ability to produce consistent and reliable study results across different assumptions, variations, or conditions [148, 174]. In our interpretation of this definition for the context of Web measurements, *robustness* implies that the outcomes of Web measurement studies should remain consistent and reliable, regardless of the natural complexity of the Web, the tools used, local regulations, or measurement strategies. The results should be consistent even when subjected to different assumptions or conditions, whether technical, geographical, or legal. This ensures that insights derived from such studies are not merely applicable to specific scenarios but can be generalized, offering reliable guidance to a broad spectrum of Web stakeholders.

We now explore ways to enhance the robustness of the studies and investigate the challenges and methods and how these approaches can contribute to enhancing the robustness of Web measurement studies.

1.2.1. Metascience: Studying Science

The history of science reveals that leading scholars have consistently sought to explore the scientific method's fundamental aspects and identify ways to implement them efficiently. Despite their solid theoretical foundations and numerous successes, it remains an ongoing challenge to determine how to apply these core principles to achieve robust results [53]. At this juncture, *Metascience*, or *the field of "studying science"*, allows us to take a step back and critically evaluate our methods and practices within research. Thus, *Metascience* deals not only with the *what* and *why* of science but also with the *how*. This reflection allows to increase the overall quality of science by studying how *repeatability*, *reproducibility*, and *replicability* can be achieved [112, 10].

In *Metascience*, researchers mainly focus on approaches to enhancing and factors that jeopardize the robustness of studies. These two aspects are addressed in detail below.

1.2.2. Enhancing the Robustness of Studies

Prominently, Ioannidis et al. [81] provide a comprehensive exploration of the core themes within *Metascience* that are pivotal for bolstering the robustness of research. We now provide an overview of these themes and subsequently interpret their relevance and application in the realm of Web measurement studies.

Methods *focus on the execution of research.* Methods encompass the basic aspects of "performing research", from study design to ethics. This area covers biases and questionable practices, emphasizing meta-analysis, evidence integration, and the importance of collaborative team science and ethical considerations.

Methods in Web Measurements

In Web measurements, "methods" refer to the systematic approaches used to gather, process, and analyze Web data. This involves selecting appropriate metrics, tools, and algorithms that can accurately capture Web activities. Addressing biases, especially those introduced by tools or the dynamic nature of the Web, becomes crucial. Collaborative efforts, perhaps in the form of open-source toolkits or platforms, can ensure consistency and ethical data collection across different studies.

Reporting *emphasizes the communication of research.* Reporting is centered around "communicating research", ensuring adherence to reporting standards and transparent disclosure of conflicts of interest. It addresses biases in reporting and dissemination, emphasizing the importance of study registration, conflict management, and methods to monitor and mitigate such issues.

Reporting in Web Measurements

For Web measurements, "reporting" emphasizes the clear and transparent presentation of data, ensuring that metrics are both understandable and actionable. This includes adhering to standards in presenting web metrics, disclosing potential conflicts (e.g., platform affiliations), and ensuring that findings are accessible not just to certain groups but also to a broader audience, including researchers, developers, and end users.

Reproducibility *targets the verification of research.* Reproducibility focuses on "verifying research" through data sharing and ensuring consistent results across repeated studies, which includes reproducibility, replicability, and repeatability. This domain tackles obstacles to data sharing, the importance of replication studies, and strategies to enhance the reproducibility and self-correction of published research.

Reproducibility in Web Measurements

In the context of Web measurements, "reproducibility" is about ensuring that measurements can be consistently re-achieved across different environments or time frames. This requires open sharing of tools, datasets, and methodologies. Overcoming obstacles like platform-specific nuances or time-bound web activities is essential to ensure that findings are not just one-off observations but can be reliably reproduced.

Evaluation *involves both pre-publication and post-publication peer review.* Evaluation revolves around "evaluating research", incorporating pre- and post-publication peer reviews and other assessment criteria. It assesses the effectiveness and costs of traditional and novel peer review methods, seeking ways to refine and improve these evaluation techniques.

Evaluation in Web Measurements

Evaluating Web measurements means assessing the methodologies and techniques used to collect and interpret Web data. Peer reviews, both prior to and after publication, ensure methodological relevance and precision. Considering conventional and innovative review approaches is vital to maintaining research quality and relevance in this dynamic domain.

Incentives *address the rewards associated with research.* Incentives pertain to "rewarding research", focusing on the criteria for promotions, rewards, and penalties in research evaluations. This area evaluates the accuracy and effectiveness of both traditional and innovative approaches to ranking and assessing the quality and value of research, teams, and institutions.

Incentives in Web Measurements

In the context of Web measurements, "incentives" relate to the recognition and rewards for producing robust, innovative, and impactful measurement methodologies. This could involve academic promotions, citations for open-source tools, badges for study artifacts or industry recognitions. Balancing these incentives with ethical considerations, especially in a rapidly evolving web landscape, is pivotal.

1.2.3. Threats to the Robustness of Studies

Certain factors inherently challenge the goal of *"Enhancing the Robustness of Studies"*. Munafò et al.'s manifesto [112], based on the framework above, reveals various factors associated with the scientific method [56] that can threaten the robustness, and thus the verifiability, of studies. These factors can be either deliberate [131, 85] or unintentional [12, 23]. Stodden et al. [153] highlight a lack of availability in research data and code, while Makel et al. [106] discuss the scarcity of replication. Kerr et al. [88] introduce the expression "HARking", describing the practice of researchers forming hypotheses based on previously known results. Simmons et al. [145] discuss the flexibility in data collection and analysis. Banerjee et al. [13] explore how to test a hypothesis, encompassing *Type I* and *Type II errors* and potential pitfalls. Other threats to reproducible science include poor study design, *P-hacking*, insufficient quality control, and publication bias [112]. In this context, multiple studies [80, 151, 26, 102] demonstrate that research findings can be incorrect or exaggerated, leading to a significant waste of research resources.

The challenges highlighted are not confined to a specific scientific domain; they are universal. Rigorous data collection and analysis are fundamental across all scientific fields. Web measurements, despite their specialization, are deeply rooted in this principle. The complexities of the Web only amplify these challenges, underscoring that Web measurements, just like procedures in any scientific discipline, must adhere to universal standards of accuracy, transparency, and reproducibility. Yet, Web measurements are an exercise in data science, and, just like any other scientific discipline, subject to the same universal challenges and demands for rigorousness, transparency, and reproducibility.

1.3. Research Questions

In light of these factors, this thesis aims to explore the application of the *Metascience* scientific method to security- and privacy-related Web measurements to identify the challenges and potential solutions within this research area. By employing metascientific exploration, we aim to identify the factors that lead to variances and, consequently, bias in Web measurements to better understand the dynamics of the Web and conduct more robust Web measurements.

This thesis primarily investigates the three major elements of *Metascience*: methods, reporting, and reproducibility. We have excluded the elements of evaluation and incentives, as these occur after a study has been conducted, as was defined in [112]. Furthermore, evaluation and incentives are largely shaped by external factors beyond the control of individual researchers. They reflect broader issues within the scientific community rather than particular problems within individual studies.

Consequently, we formulate the central research question of our study as follows:

RQ1 Robustness ?

How can we ensure the robustness of Web measurement studies, and which factors lead to variances in security- and privacy-related Web measurements?

Our objective is to pinpoint factors that lead to variances in security- and privacy-related Web measurements and to create more robust measurement methods. This is fundamentally important and serves as a basis for reliable, trustworthy, and reproducible research through robust methods. We can achieve more reliable results by gaining a better understanding and control of the factors causing these variances [112, 81]. These enhanced measurements can capture challenges more accurately and provide more precise insights and actionable recommendations for the stakeholders.

From this consideration, we derive the second research question, which serves as a case study, drawing upon the findings of RQ1 and illustrating the practical implications of these refined measurement techniques within the Web security domain:

RQ2 Security ?

Having the findings from RQ1, how can we conduct Web security measurements, as a case study to understand the update behavior and vulnerability of websites?

Understanding privacy on the Web is as crucial as security. Robust and reliable Web measurements help understand the practices and mechanisms of privacy protection, thus facilitating improvements in these practices. In light of these aspects, our investigation leads to the third research question:

RQ3 Privacy ?

Having the findings from RQ1, how can we conduct Web privacy measurements, as a case study to understand the effectiveness and impact of privacy tools?

By integrating these three research questions, we propose a comprehensive approach to enhancing Web security and privacy by applying the principles of *Metascience* to Web

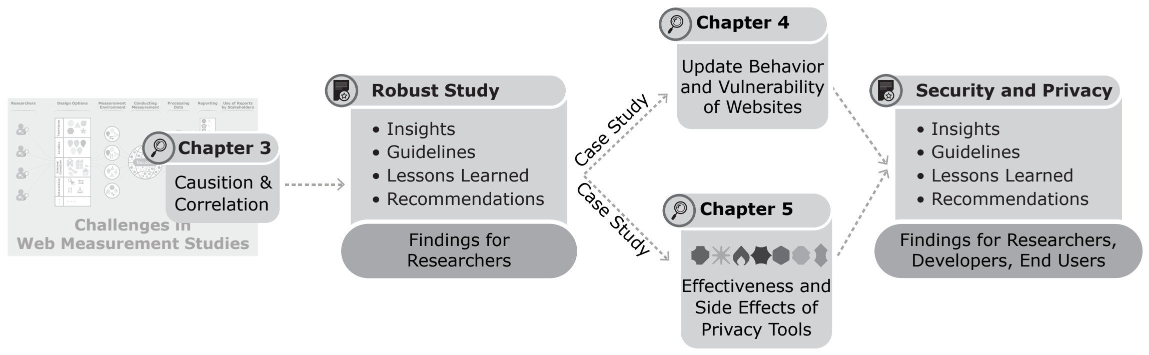

Figure 1.2.: A visual overview of contributions of this thesis.

measurements. Based on our findings, we aim to provide robust results for key stakeholders. In this work, we identify three key target groups that stand to benefit significantly from robust Web measurements:

Researchers — *Investigators of the Web.* Web crawlers have become increasingly significant in modern research, with 16% of studies from popular scientific conferences and symposiums on computer networks utilizing web crawler techniques to obtain data and provide insights for stakeholders such as other researchers, developers, and end users [4]. However, due to the complexity of the Web, measurement variances and reproducibility and replicability often occur, making it essential to explore ways to ensure that Web measurements are reproducible and replicable and provide robust results.

Developers — *Providers of the Web.* As providers of the Web, developers, and operators play a key role in the overall security and privacy of the Web and are assigned major responsibility in this respect. Consequently, it is essential to investigate the extent of their platforms' resilience to impacts on security and privacy.

End users — *Consumers of the Web.* Currently, 63% of the global population actively engage with the Web, a percentage that continues to grow [157]. This widespread usage makes ensuring and assessing software components' security and privacy increasingly vital, guided by robust findings from reliable studies.

1.4. Contributions

The primary contributions of this thesis guide researchers in performing robust Web measurement studies. Drawing on our findings, we conduct two Web security and privacy case studies to offer insights, practical guidelines, and actionable recommendations for researchers, developers, and end users through multiple large-scale measurements.

In this thesis, we executed multiple comprehensive large-scale Web measurements utilizing 74 distinct measurement profiles (i.e., various web browsers, geographic locations, and measurement methodologies) resulting in the crawling of six million sites and 17 million webpages across the global Web. Our research covers various components of the Web, providing an evaluation to understand the dynamic nature of the Web and the factors contributing to its complexity. The methodologies and evaluations enhance our understanding and facilitate the generation of robust, replicable, and reproducible Web measurements. Based on the findings of our measurements, we provide security- and privacy-related insights and propose guidelines and recommendations for researchers, developers, and end users alike. The contributions of this thesis are outlined below. An overview of them is also given in Figure 1.2.

- **Robustness of Web Measurements** — *Enhancing robustness of Web measurement studies.* We developed an open-source framework, *MultiCrawl* [1], which enables the design of comprehensive Web measurement setups. Utilizing this framework, we conducted two large-scale Web measurements to identify the challenges when designing reproducible and replicable studies. In the first study, of 4.5 million pages using 24 different profiles, our findings indicate that even minor modifications in the measurement design directly affect the results, leading to observable differences in the collected data. In a second study with five different profiles on 1.7 million webpages, we want to understand *where*, *how*, and *why* differences arise when visiting webpages. Thus, we identify the characteristics (e.g., tracking, third-party inclusions, or content types) of resources within websites that contribute to increased dynamism and, consequently, variations. We provide recommendations and guidelines for enhancing robustness of Web measurements (Chapter 3).
- **Security of the Web** — *Understanding vulnerability of the software components.* In the second part of this thesis, we delve into the security aspects of the Web. We analyze large-scale measurements made on 5.6 million websites over 18 months and examine 246 distinct server-side and client-side software distributions. We demonstrate that nearly all examined websites (95%) employ at least one software component with a known vulnerability, and the number of vulnerable websites increases over time. It is also found that 96% of these websites employ at least one outdated software component which is often older than four years. We finally provide insights, guidelines, and actionable recommendations to enhance Web security (Chapter 4).
- **Privacy of the Web** — *Understanding the efficiency of privacy tools.* Lastly, we shift our focus to the privacy aspects on the Web through further large-scale measurements, involving 2.5 million webpages with 27 different profiles, to evaluate the effectiveness of "cookie banner interaction" tools. In these measurements, we examine the effectiveness of "consent banner interaction" extensions to enhance Web privacy for the stakeholders. We observe notable differences even among extensions with the same features and capabilities (e.g., accepting functional cookies), show how these extensions interact with consent banners in distinct ways, and provide insights, guidelines, and recommendations to enhance the efficiency of developing and deploying these tools (Chapter 5).

[1] `https://github.com/nrllh/multicrawl`

1.5. Impact

Within the scope of the studies conducted for this thesis, we furnish several insights into enhancing the robustness of Web measurements and reinforcing the security and privacy of the Web for different stakeholders. To ensure the use of our findings to researchers, developers, and end users, we translate them into pragmatic guidelines and actionable recommendations. Thus, we strive to make an impact on strengthening the robustness of the Web's security and privacy, thereby creating a more secure and privacy-enhanced environment for its users. In the following, we outline the potential impact of our findings on the aforementioned stakeholders and elucidate how they will benefit from our research:

Researchers. This thesis's methodologies, techniques, and findings serve as a robust foundation for future Web security and privacy research. They provide a roadmap for designing and conducting large-scale Web measurements to enhance robustness. These insights could prompt a broader adoption of robust measurement methods, leading to more reliable results and, subsequently, more accurate interpretations and conclusions.

Developers. The prevalence of outdated and vulnerable software components underlines the need for more efficient software update mechanisms. Hence, developers can leverage these insights to introduce new strategies for enhancing the efficiency of software updates and improving overall Web security. Moreover, our evaluation of various privacy tools can guide developers in creating more effective and efficient privacy-enhancing solutions, thereby contributing to enhanced privacy on the Web.

End users. The results of the studies reflect the current state of vulnerabilities on the Web and the impact and side effects of privacy tools for end users, thus raising their awareness. Based on our actionable recommendations, users can navigate the Web more securely and make more informed decisions about the tools they use to protect their privacy.

We hope this effort will contribute to enhancing overall security and privacy on the Web.

1.6. Thesis Overview

The following chapters deal with large-scale Web measurements to improve their *robustness* within the contexts of *security* and *privacy*.

Chapter 2 lays the foundation for understanding Web measurements, introducing the concepts and practices involved. We define the key terminology used in Web measurement studies. We then discuss the methodologies and techniques for designing robust Web measurement studies. We explore different Web security and privacy aspects. This chapter provides a holistic view of the elements utilized in this thesis and considerations in conducting Web measurement studies.

Chapter 3 focuses on the enhancing robustness aspects of Web measurement studies. We start by addressing the robustness of Web measurement studies, emphasizing the importance of consistent methodologies and procedures for ensuring the reliability of research findings. Next, we explore the similarity of Web privacy measurements and examine the factors that contribute to the comparability of results across different studies. This chapter aims to provide a comprehensive understanding of the design considerations necessary for conducting robust Web measurement studies to draw reproducible, replicable, and verifiable conclusions.

Chapter 4 shifts our focus to security by specifically examining the update behavior of websites and its impact on security. We investigate the factors influencing update practices and the consequences of outdated software components for Web security. By analyzing real-world data, we aim to provide insights into the relationship between website maintenance and security vulnerabilities and provide recommendations for researchers, developers, and end users. This case study shows that static open-source data, such as *HTTPArchive*, ensures fully reproducible Web measurement results.

Chapter 5 concentrates on privacy and studies consent banner interaction tools and their impacts on user privacy. We analyze the effectiveness and, efficiency, and consequences of various consent banner interaction tools. Through this in-depth investigation, we aim to provide insights into the role of such tools in Web privacy and provide recommendations for researchers, developers, and end users based on our findings. Additionally, this Chapter illustrates the variance in results of the privacy tools with similar functionalities.

Chapter 6 concludes the thesis by highlighting the implications of our research, and briefly discusses potential avenues for future exploration.

Appendix A provides references to our code, raw measurement data, queries for the entire data processing pipeline and evaluation, and other supplementary materials of all presented works in this thesis.

1.7. Publications

This thesis is based on several peer-reviewed academic papers published at various competitive research conferences. We explore the interconnectedness of these papers and the ensuing chapters. This highlights also my contributions to these collective works.

Chapter 3 is primarily based on two papers. The first, "Reproducibility and Replicability of Web Measurement Studies", was co-authored with Matteo Große-Kampmann, Tobias Urban, Christian Wressnegger, Thorsten Holz, and Norbert Pohlmann and published at *The Web Conference 2022* (WWW 2022). I was responsible for the design of the measurement framework and the evaluation of the measurement data, in collaboration with Tobias Urban. Matteo Große-Kampmann contributed the literature review to the paper, and he included the relevant results in his doctoral thesis. Therefore, the results related to the literature review are not included in this thesis. Christian and Thorsten supervised the paper's finalization. The second paper, "On the Similarity of Web Measurements Under Different

Experimental Setups", was written in collaboration with Jan Hörnemann, Matteo Große-Kampmann, Tobias Urban, Norbert Pohlmann, Thorsten Holz, and Christian Wressnegger. This paper stands to be published at the *Internet Measurement Conference* (IMC 2023). I made the measurements and conducted the complete data analysis with contributions of Jan Hörnemann and having in-depth discussions with Tobias Urban. The paper was finalized by Matteo Große-Kampmann, Thorsten Holz, and Christian Wressnegger.

Chapter 4 relies on the paper "Our (in)Secure Web: Understanding Update Behavior of Websites and Its Impact on Security". This publication, a joint effort with Tobias Urban, Kevin Wittek, and Norbert Pohlmann, was presented at the *Passive and Active Measurement Conference* (PAM 2021). I was responsible for carrying out the entire measurement procedure and appraising the resulting data, guided by Tobias Urban. The finalization of the paper was handled by Tobias Urban.

Finally, the paper "A Large-Scale Study of Cookie Banner Interaction Tools and Their Impact on Users' Privacy" serves as a basis for Chapter 5. This work, co-authored with Tobias Urban, Norbert Pohlmann, and Christian Wressnegger, is slated for presentation at the *24th Privacy Enhancing Technologies Symposium* (PETS 2024). My role was to lead the design of the measurement framework and analyze the results. The paper was finalized by Tobias Urban and Christian Wressnegger.

These papers form the foundation of this thesis:

- Reproducibility and Replicability of Web Measurement Studies
  Nurullah Demir, Matteo Große-Kampmann, Tobias Urban, Christian Wressnegger, Thorsten Holz, and Norbert Pohlmann
  In *Proc. of 31st ACM Web Conference 2022 (WWW)* [35]
  Best paper award candidate
- On the Similarity of Web Measurements Under Different Experimental Setups
  Nurullah Demir, Jan Hörnemann, Matteo Große-Kampmann, Tobias Urban, Norbert Pohlmann, Thorsten Holz, and Christian Wressnegger
  In *Proc. of the 23rd Internet Measurement Conference 2023* (IMC) [36]
- Our (in)Secure Web: Understanding Update Behavior of Websites and Its Impact on Security
  Nurullah Demir, Tobias Urban, Kevin Wittek, and Norbert Pohlmann
  In *Proc. of 22nd Passive and Active Network Measurement 2021* (PAM) [39]
- A Large-Scale Study of Cookie Banner Interaction Tools and Their Impact on Users' Privacy
  Nurullah Demir, Tobias Urban, Norbert Pohlmann, and Christian Wressnegger
  In *the 23rd Privacy Enhancing Technologies Symposium 2024* (PETS) [38]

Additionally, several publications (e.g., book chapters, peer-reviewed papers) were produced during my doctoral study, although they are not incorporated into this thesis.

- Angriffe auf die Künstliche Intelligenz – Bedrohungen und Schutzmaßnahmen
  Dominik Adler, Nurullah Demir, Norbert Pohlmann
  In *Proc. of DATAKONTEXT-Fachverlag*. 2023 [2]
- Towards Understanding First-Party Cookie Tracking in the Field
  Nurullah Demir, Daniel Theis, Tobias Urban, and Norbert Pohlmann
  In *Proc. of GI Sicherheit*. 2022 [37]
- The 2022 Web Almanac: HTTP Archive's annual state of the web report: Privacy
  Tom Van Goethem, Nurullah Demir
  In *HTTPArchive*. 2022 [166]
- The 2021 Web Almanac: HTTP Archive's annual state of the web report: Security
  Saptak Sengupta, Tom Van Goethem, and Nurullah Demir
  In *HTTPArchive*. 2021 [140]
- The 2020 Web Almanac: HTTP Archive's annual state of the web report: Security
  Tom Van Goethem, Nurullah Demir, and Barry Pollard
  In *HTTPArchive*. 2020 [167]

Chapter 2

Background on Understanding Web Measurements

This chapter offers an overview of the theoretical foundations of the research presented in this thesis. We begin with an introduction to key terminology in Section 2.1, followed by a discussion of the design considerations of Web measurement studies in Section 2.2. In Section 2.3, we explore the representation of websites as trees. We then explore the importance of understanding Web technologies and updating practices in the context of vulnerabilities in Section 2.4 and Section 2.5. Lastly, we examine the role of cookies in Section 2.6 and the use of cookie banners for tracking and consent management in Section 2.7.

2.1. Terminology

We start by introducing key terminology. In this thesis, we use the term *site* (or *website*) to describe a registerable domain, sometimes referred to as *eTLD+1* ("extended Top Level Domain plus one"). Examples for sites are `foo.com` and `bar.co.nz`. Each site may have several *subdomains* (e.g., `news.foo.com` and `sport.foo.com`). Following the definition of RFC 6454 [15], we call the tuple of protocol (e.g., HTTPS), subdomain (or hostname), and port *origin*. This distinction is important since the well-known security concept *Same-Origin Policy* (SOP) guarantees that pages of different origins cannot access each other. We use the term *page* (or *webpage*) to describe a single HTML file hosted at a specific URL (e.g., `https://news.foo.com/bar.html`).

2.2. Designing Web Measurement Studies

Web measurements are indispensable for understanding the web, its adherence to legislation, and the functioning of its vast ecosystems [162, 161, 39]. They illuminate web content trends and the extent of vulnerabilities in technologies such as TLS [132, 11, 110], and infrastructural aspects as SSL or HTTP/3 adoption [160, 16].

However, the complexity and dynamism of the web make these measurements challenging. The dynamic content display mechanisms and variations in technology stacks complicate web measurements [25]. Prior work has shown that slight setup changes can significantly influence the results, emphasizing the importance of understanding how various factors affect the outcomes [34]. The provided examples highlight that conducting Web measurements is complex, and previous studies have already shown that the outcome of an experiment is affected by several factors (cf. Section 1.1). This thesis aims to fill a research gap by examining the structure and dependencies of dynamic content.

2.3. Representing Websites as a Tree

With their overall complexity and dynamic content, websites necessitate a uniform representation to conduct systematic analysis. One efficient representation is a tree that models all resources and their dependencies [161, 75, 146]. We build trees following previous work to enhance our thesis's comparability. The edges in a tree symbolize HTTP communication, and loaded content represent a node. Thus, if an element on a page loads additional content, such as images (i.e., nodes), it will trigger HTTP requests (i.e., edges). This approach is used in one of our study (cf. Section 3.2).

2.4. Web Technologies & Updating

To implement modern web applications, service providers rely on a diverse set of server-side (e.g., PHP or MySQL) and client-side technologies (e.g., HTML or JavaScript). This combination of different technologies often results in a very complex and dynamic architecture, not always under full control of the service provider (e.g., usage of third parties [75]). Furthermore, the update frequency of web technologies is higher compared to desktop software [158]. Web applications are commonly composed of different modules that rely on each other to perform a given task. Hence, one vulnerability in any of these modules might undermine the security of the entire web app, depending on the severity of the vulnerability. Once a vulnerability of an application is publicly known or privately reported to the developers (see also Section 2.5), the provider of that application (hopefully) provides an update to fix it. Therefore, service providers need to check the availability of updates of the used components and their dependencies and transitive dependencies regularly. However, it should be noted that not all updates fix security issues, and, therefore, it is not necessary or desired (e.g., for stability reasons) to install all updates right away.

2.5. Common Vulnerabilities and Exposures

Once vulnerabilities in software systems are discovered, reported to a vendor, or shared with the internet community publicly, they are published in vulnerability database platforms (e.g., in the *National Vulnerability Database* (NVD)). The NVD utilizes the standardized *Common Vulnerabilities and Exposures* (CVE) data format and enriches this data. Each CVE entry is provided in a machine-readable format and contains details regarding the vulnerability (e.g., vulnerability type, vulnerability severity, affected software, and version(s)). The primary purpose of each CVE entry is to determine which software is affected by a vulnerability and helps to estimate its consequences. Each entry in the NVD database is composed of several data fields, of which we now describe the one most important for our study in Chapter 4.

In the NVD database, the field `ID` of a CVE entry uniquely identifies the entry and also states the year when the vulnerability was made public, followed by a sequence number (e.g., `CVE-2020-2883`), the field `CVE_data_timestamp` indicates when the CVE entry was created. Furthermore, each CVE entry also includes a list of known software configurations that are affected by the vulnerability (field `configurations`), formally known as *Common Platform Enumeration* (CPE). CPE defines a naming scheme to identify products by combining, amongst other values, the vendor, product name, and version. For example the CPE (in version 2.3) `cpe:2.3:a:nodejs:node.js:4.0.0:[...]` identifies the product `node.js` provided by the vendor *nodejs* in version 4.0.0. Furthermore, the `configurations` field lists all conditions under which the given vulnerability can be exploited (e.g., combination of used products). Finally, the field `impact` describes the practical implications of the vulnerability (e.g., a description of the attack vector) and holds a score, the *Common Vulnerability Scoring System* (CVSS), ranging from 0 to 10, which indicates the severity of the CVE (with ten being the most severe). Again, it is worth noting that it not definite that if one uses a software product – for which a vulnerability exists – that it is exploitable by an attacker. For example, if an SQL-Injection is possible via the comment function of a blog, it can only be exploited if the comment function is enabled. Thus, our results can be seen as an upper bound.

2.6. Cookies

A *cookie* is a piece of text communicated via an HTTP header or set via JavaScript. More specifically, a cookie is a key-value pair set on a client by a visited website or third-party present on that website. Cookies were implemented to allow stateful communication in the otherwise stateless HTTP protocol [14]. Therefore, they are often used to manage sessions, store persistent client-side data, and often for user tracking purposes [161]. Various organizations proposed categories of cookies to account for the wide range of use cases and to bring some transparency to the usage of cookies(e.g., the *IAB Europe* proposed 17 categories of cookies [74]).

2.7. Cookie Banners

Websites often display so-called “cookie banners” that inform users about the usage of cookies and sometimes provide them some level of control over the type and amount of cookies a website may set. In the field, three types of such banners have been established [163, 33]: (1) banners that only inform users that a site uses cookies without control, (2) banners that allow users to accept or reject all cookies, and (3) banners that provide fine-grain control of different types of cookies a page may set. Different services have emerged that help website providers manage the communication of consent. Such services are often referred to as “consent management platforms” (CMP) [176, 66, 92]. In a nutshell, CMPs are standard “off-the-shelf” software that aim to automate the consent management process for websites.

Chapter 3

Robustness: Designing Web Measurement Studies

Given the complexity of the Web, it is challenging to reproduce or even replicate the results of Web measurements. Many factors directly influence their reproducibility and replicability. However, reproducibility and replicability of a scientific study are essential.

In this Chapter, it is discussed how to enhance the reproducibility and replicability of Web measurements. We first examine the circumstances under which the results of Web measurements vary and then identify *where*, *how*, and *why* this variance is observed. Thus, this Chapter consists of two related comprehensive studies. In the first study (cf. Section 3.1), we discuss challenges and best practices in designing Web measurements and present a set of criteria for researchers to enhance the robustness of their studies. Subsequently, we conduct a comprehensive Web measurement to verify these criteria and understand their effect on the robustness of the measurements. In the second study (cf. Section 3.2), we delve deeper into our analysis by conducting another measurement with multiple profiles to investigate the precise reasons for the differing results in Web measurements. For this, we construct trees for each page per profile and compare these trees. This analysis helps us better understand the complexity of the Web, thus allowing us to take measures to enhance the robustness of Web measurements.[1]

3.1. Reproducibility and Replicability of Web Measurement Studies

As the Web has grown to an essential part of our day-to-day life, the complexity of the employed web applications has increased drastically. This development has been accompanied by undesirable practices, such as user tracking [162, 40, 83], fingerprinting [122, 44],

[1] The remainder of this Chapter is mainly based on two published papers. The first, "Reproducibility and Replicability of Web Measurement Studies", was showcased in the 31st ACM Web Conference 2022 (WWW) proceedings. This work was co-authored by Matteo Große-Kampmann, Tobias Urban, Christian Wressnegger, Thorsten Holz, and Norbert Pohlmann [34]. The second paper, "On the Similarity of Web Privacy Measurements Under Different Experimental Setups", stands to be presented in the proceedings of the 23rd Internet Measurement Conference 2023 (IMC) and co-authored by Jan Hörnemann, Matteo Große-Kampmann, Tobias Urban, Norbert Pohlmann, Thorsten Holz, and Christian Wressnegger and presented in this thesis with additional analyses [36]. It is dedicated to Tarık.

or even outright malicious activities, such as XSS attacks [178]. Web measurement studies are an essential tool to understand, identify, and quantify such threats, and they allow us to explore isolated phenomenons at a large scale. As the modern Web is highly dynamic and ever-changing, this is an inherently difficult task. To conduct studies across thousands of websites, researchers can partly rely on crawling frameworks such as *OpenWPM* [44], but more often, they have to extend existing work or build new crawlers on their own to adapt to new developments on the Web.

This trend, however, raises the question of whether different measurement studies using different frameworks for gathering data are comparable *and* to which extent experiments can be reproduced or replicated. In particular, in the field of Web-based measurements, ensuring replicability requires a tremendous effort to describe, document, and openly communicate the details of the experimental setup and implementations. However, if the community cannot verify and reenact drawn conclusions, the entire scientific process is at risk of becoming unreliable—something that has unfortunately been observed in different research disciplines in the past [79, 144, 31, 114].

In this section, we systematize such effects, provide best practices and criteria that help design studies, and additionally perform a large-scale Web measurement study that highlights the impact of these subtle differences. We define criteria that help designing experimental setups that are reproducible and replicable. In a large-scale study for which we visit 4.5 million pages on over 8,800 sites with 24 browser profiles, we show that slight changes in the experimental setup alters the results to an extent where cross-comparability of studies is not feasible (see Section 3.1.2). For example, we find that the identified trackers on pages can vary by 25% based on the used browser configuration.

In summary, we make the following contributions:

- **Guidelines for robust Web measurements.** We highlight the challenges of designing Web measurements and provide guidelines that help setting up experiments that effectively address them.
- **MutliCrawl.** We provide an open-source framework to conduct comprehensive measurements, accommodating various profiles with configurations such as different locations, browsers, and interactions.
- **Impact analysis.** To increase the comparability of future and previous Web measurements, we perform experiments utilizing 24 measurement setups and compare the measured differences that emerge from the utilized frameworks. We also emphasize factors threatening the robustness of Web measurements.

3.1.1. Method on Designing Web Measurement Studies

The rapidly changing, variable content and the general trend towards providing more content online makes the Web a challenging subject for measurement studies. As an example, suppose one visits the same website at the same time with different browser instances. The loaded content (e.g., ads or other dynamic content) likely differs and, thus, the overall results of measurement studies might deviate (e.g., when identifying embedded trackers or analyzing shown ads). This simple example illustrates that repeated experiments may show (slightly) different results and conducting such experiments in an uncontrolled environment is bristled with obstacles, endangering replicability. However, the cornerstone of academic work is the possibility to scrutinize conclusions and results. We thus pick up the definitions of the Association for Computing Machinery [10] for a) repeatability (*"Same team, same experimental setup"*), b) reproducibility (*"Different team, same experimental setup"*), and c) replicability (*"Different team, different experimental setup"*).

We put a particular emphasis on reproducibility and replicability of published studies, and leave repeatability aside, as by definition it can only be achieved by the team that conducted the experiment in the first place. Thus, reproducibility and replicability are essential to our analysis, as these enable us to verify and compare results of existing work.

3.1.1.1. Challenges and Best Practices

In the following, we present design patterns and best practices that help to plan Web measurement studies so that future studies can be designed to be reproducible and replicable. We create these guidelines based on the surveyed literature and our own experience in this research area. For all surveyed papers, we analyze the documented setup of each experiment, abstract general design choices, and develop best practices that are intended to provide an overview of aspects which need to be considered when designing Web measurements in practice. It is essential to highlight that our guidelines are not intended to point fingers or criticize previous work, but to highlight pitfalls and challenges that can impact a study's outcome.

Method to Design the Best Practices. To derive the best practices, we analyze different experiment design choices of the papers and compare the outcomes of the works. This allows us to identify generalizable and common aspects that are shared across different works. For example, if one work visits sub-pages and another work only visits landing pages but both find different levels of tracking activities, we create a best practices that researchers should take this behavior into account. Moreover, we use these best practices to derive criteria that measurement studies should follow to allow for reproducibility of conducted experiments.

Building the Dataset. Naturally, each Web measurement study has to identify websites and pages to be analyzed during the experiment. For this step, one can distinguish between three methodically different approaches, which all come with up- and downsides.

P1 **Artificially selecting websites and pages.** As the Web is ever-growing and consists of a myriad of sites with even more pages, measuring all of them in a single experiment is not feasible in a reasonable way. A commonly accepted way of focusing an experiment is to use a so-called "top-list" that ranks popular sites (e.g., *Alexa* [5], *Tranco* [100], or others [8, 105]). These lists, however, only include the landing page (or the eTLD+1) which are used for the experiment. While, at first glance, this might seem reasonable, recent works have shown that sub-sites (e.g., *https://www.example.com/news*) show a significantly different behavior than the respective landing pages [161, 8], and that the rank of a website also might impact the results [179]. Consequently, we advocate to name the sources (e.g., top-list) of sites that have been analyzed, detail how they have been picked, and list all analyzed pages (e.g., in an appendix). Similar to the highlighted challenges to enable repeatability, it is necessary to point out which criteria are used to choose or eliminate entries from a given set of sites.

P2 **Using user clickstream data.** Another approach is to use clickstreams observed from real users or analyze their traffic directly. While such an approach is more realistic, they are harder to collect. However, studies that explicitly need to understand the effects of a phenomenon for individual users need to take this step [17, 121, 49]. If 'only' the presence of a phenomenon is of interest (e.g., if secure CSPs are used), artificially selected sites can suit the purpose.

P3 **Use existing sources.** Using a previously collected public datasets (e.g., *HTTPArchive* [70]), is the only option that allows the reproduction of results, offers high repeatability, and enables to compare properties. However, one is bound to analyze phenomena for which data is already present in the desired granularity [39], which is often not the case.

From this set of best practices, we derive four criteria (C1–C4) that a measurement study should meet (see group *"Dataset"* in Table 3.1). While criteria C1–C3 (*"documentation of the analyzed sites"*) are directly related to the named practices, criterion C4 is intended to highlight that some phenomena need to be analyzed over time to understand their scale, we analyze its effects in Section 3.1.2.

Experiment Design. One way or another, Web measurement studies rely on a crawler. Selecting, building, and customizing such a crawler is an essential step in preparing each study, such that one needs to prudently design and implement it to ensure that the experiment is stable, repeatable, and comparable.

Building the Crawler. We now discuss design decisions when performing a study using artificial browsing data (i.e., not using user-generated or public data). We review the essential steps that should be taken into account when designing such a study:

P4 **Choosing a technology.** Previous work relies on different measurement setups ranging from simple tools like *cURL* [29] to sophisticated measurement frameworks that can spawn several browsers at once like *OpenWPM* [111, 44]. As prior work has shown, the decision of which tool to use impacts the results [4].

P5 **Customization of the crawler.** Naturally, each study uses a (slightly) different measurement setup. When customizing a crawler, it is inevitable to elaborate on the steps taken and discuss possible artifacts and limitations of the approach. While necessary, each customization step might impact the results (e.g., different user agents) and, therefore, needs to be documented [104]. We discuss these effects in more detail in Section 3.1.2.

P6 **Avoiding crawler detection.** Crawlers and other bots make up approx. 37% of traffic on the Web [76] and it has been shown that this significantly affects crawling studies [170, 78, 107]. Consequently, some service providers define behavior guidelines to limit crawling traffic, or try to detect and block them altogether [86]. These defense mechanisms might substantially impact the results of measurement studies if sites present different content or none at all. Hence, the authors' choice to avoid and if so how and to which extent an evasion technique was implemented needs to be discussed transparently. However, it is commonly accepted (and often necessary) to circumvent bot detection mechanisms [161, 162, 44].

P7 **Mimicking User Interaction.** Modern websites are no longer static HTML pages, but interactive applications that load different sets of content depending on the users' actions. Resources are often only loaded once visible to the user (known as "lazy loading") to improve the website's loading speed and for search engine optimization purposes [62]. This means that crawlers that do not interact with a page (e.g., scrolling) will miss crucial resources [161, 162, 87, 180]. Therefore, interaction mechanisms need to be documented, and limitations of lacking user interaction should be discussed.

Based on these four aspects of a crawling setup, we derive criteria C5, C6, C7, C8, and C10. We split the customization step (P5) into two criteria (C6 & C7) to account for differences whether a crawler was modified (e.g., a function was altered) or extended (e.g., a browser extension was used). Furthermore, we add a criterion that urges authors to make the crawler publicly available (C9). Since the effects of C5 and C11 are not yet adequately discussed by previous work, we analyze them in Section 3.1.2.

Experiment Environment. After selecting the sites to visit and building the crawler, the experimental environment must be crafted. In the following, we describe essential environmental aspects that may impact the crawler and, therefore, the experiment's outcome.

P8 **Geolocation of crawls.** A critical factor for each experiment is the location from which the measurement study is conducted. Depending on the location (e.g., based on the IP address of the crawling machine), websites might deliver different content [72]. This may, for instance, be founded in cloaking, legislation (e.g., the GDPR or CCPA [179, 162, 30, 161]), or even censorship [117, 21]. Such impacts have to be accounted for (e.g., via using a VPN setup) and actions to address them need to be disclosed in detail.

P9 **Defining the page visit strategy.** For the page visit strategy, we distinguish between *stateless* and *stateful* crawls. A stateless crawler (i.e., browser) is reset completely between each page visit, such that each visit creates a new HTTP session that updates the browser's internal resources. On the contrary, some (e.g., only the cookie jar) or all of this information is kept in stateful crawls, as a "real" browser would. Consequently, authors need to document what part of a browser profile is maintained statefully, what part is reset, and when [44, 180]. This distinction has a severe impact on the outcome of the experiment: In stateful experiments, the order of visited pages potentially impacts the results, and it accounts for HTTP session-specific phenomena, such as opt-in to cookie tracking. Stateless crawls, in turn, allow to study session-independent attributes. Note that this practice does *not* account for browser profiles that were populated before the measurement took place (e.g., by pre-filling the cookie jar), we account for this in P11.

P10 **Setting up Browser Configuration.** A browser' configuration plays an important role for Web-based measurements. Depending on the browser (e.g., version) the crawled entities might act differently. To allow comparability and reproducibility of experiments, it is essential to share basic configuration details, which may impact the study's outcomes [98, 169, 180]. Such design choices range from installed extensions, used block-lists, login strategies, used browser version, content of the cookie jar, etcetera.

P11 **Describe shortcomings and limitations.** Naturally, a Web measurement can never be complete regarding, for instance, coverage or realism. The experimental design accounts for these "natural" boundaries, but each design choice will likely impose certain restrictions and limitations. To allow the research community to acknowledge and assess the outcomes of an experiment fully, it is inevitable to discuss the limitations of its design [133, 128].

From these practices, each can be mapped to a single criterion (C11, C12, C13, and C17). We add an additional criterion (C15) asking to make results publicly available, as this particularly helps to replicate or reproduce an experiment. Moreover, we set up two criteria that help assessing the findings of a paper: First, asking for an ethical discussion (C18) and second, urging to provide a general overview of the measured results (C16).

3.1.1.2. Design and Evaluation Criteria

Based on the best practices described in the previous section, we derived the named 18 criteria to allow reproducibility and replicability of a study. In a first step, two experts, both with an extensive professional and academic background in security and privacy on the Web, assessed an identical, randomly selected subset of the surveyed papers ($n = 25$) to test the applicability of the criteria. This exploratory evaluation has shown a very high interrater reliability (Cohen's kappa: $\kappa = 0.94$), which indicates that the designed criteria can be unambiguously applied. In a few cases, the experts have disagreed, which however turned out to be founded in an initially ambiguous formulation of one criteria, which was adjusted accordingly. In a second step, the criteria have then been applied to 117 papers in our corpus [34]. The results of this survey are included in one of the co-author's doctoral thesis. Table 3.1 lists all 18 criteria and provides a brief description of each.

3.1.2. Results

In this section, we proceed to demonstrate the impact of insufficiently documented experimental setups of large-scale studies along four exemplarily case studies focusing on C4, C5, C10, and C12. The first three are chosen because the literature currently does not provide enough evidence on their impact, while C12 is used to verify that our framework is able to reproduce previous results.

Table 3.1.: Criteria to design Web measurement studies.

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

3.1.2.1. Web Measurement Approach

To show the impact of seemingly small changes in a measurement setup on the replicability of an experiment, we perform a Web measurement study that uses 25 different setups. More specifically, we compare results of four browsers (*Firefox*, *Firefox headless*, *Chrome*, and *Chrome headless* – C5), three regions (Europe, Asia, and North America – C12), and two types of website interactions (*"none"* and *"simple interaction"* – C10) individually. Overall, we cross-compare 24 different setups. Additionally, we perform a repeating study that measures the same sites and pages on a daily basis – C4.

The data corpus of our study consists of the top 10k Tranco [100] websites and we collect the first 25 subpages for each site (as identified by the JavaScript engine), if possible. We designed a pipeline that coordinates all page visits across the profiles. Our measurement setup consists of virtual machines (VMs) orchestrated by a "commander" instance to organize parallel page visits. For example, one VM performs the measurement using Chrome from the US with user interaction, while another does the same for the EU. A detailed description of our framework can be found in Appendix A.2.1.

The commander is in charge of starting the measurement for each site in parallel across the VMs. Each VM starts 10 browsers (one for each site) in parallel using the defined profile. Once the analysis of a page is finished, the same browser instance is moving to the next page of the same site. Hence, subsequent visits of pages will not be synchronized across all VMs. On the landing page level, the timing differences in our experiment are only 17 seconds on average. However, the timing differences on subpage level are 3 min (SD: 7 min). When visiting a page, each browser logs all HTTP requests and responses and stores them in a central database. We wait until a page has finished loading or a timeout of 30 seconds is reached, close the browser, and move on to the next page.

3.1.2.2. Replicability of Measurements

We highlight the impact of individual criteria based on four examples and explore them along two dimensions: (1) Web tracking and (2) usage of *Content Security Policies* by a page.

Method. To compare the results of the 24 profiles, we use the *Jaccard index*. For each page, we have a set of observed trackers and CSPs (i.e., 24 sets). The Jaccard index is used to gauge the similarity of sets. The index computes the similarity by dividing the size of the intersections with the size of the union of all sets. By design, the index ranges from 0 to 1, where 1 denotes that the sets are equal and 0 indicates that they have no element in common. This allows us to compare and quantify the differences in observed trackers on page level across all profiles. The Jaccard index is used to compare the similarity of two sets. Since we compare multiple sets, we compute the pairwise similarity between all sets and use the arithmetic mean to state the similarity for a given page.

We analyze the impact on privacy-related studies along with the presence of trackers. More specifically, we analyze which tracking requests are observable when visiting a page.

To identify them, we use the tracking filter list *EasyList* (as off 07/05/2021) [42]), which we provide in the supplementary data of this work (see Section A.1). If an observed URL is present on the list, we consider it to be a tracking request. Furthermore, we use the eTLD+1 part of these URLs to identify *trackers*, in terms of domain names.

To get a better understanding of the impact of different measurement setups for security studies, we analyze the presence of *Content Security Policies* (CSP). CSPs help to mitigate specific attack vectors on the Web (e.g., XSS attacks). They are implemented by an HTTP header that contains different directives that define sources from which content may be loaded. We analyze differences in CSPs by inspecting the used CSP directives and all attributes within a directive. We omit all variable attributes (e.g., nonces) in the analysis since they change by design. Furthermore, we compare the semantic effect of a directive (i.e., ordering is ignored).

General Measurement Overview. Our total website corpus consists of 10k distinct sites and we found 182,586 subpages on those sites, including the landing pages. Across all profiles, we successfully visited 4.5M pages on 8,883 sites. The sites that could not be crawled are not meant to be visited by a human (e.g., link shorteners, content delivery networks, or ad networks). The resulting database has a size of roughly 1.1 TB, which is openly available (see Section A.1). On average, each profile visited 179,404 pages (SD: 6,947; max: 186,972; min: 158,691). In our analysis, we only consider pages for which we observed at least 17 successful crawls across the 24 profiles. Hence, roughly 70% of the profiles have to visit a page so that we consider it. Furthermore, this guarantees that at least one profile in each category successfully crawled a page. 178,452 (92%) of the analyzed pages fall into this category. Note that 134,120 (75%) of pages were successfully crawled by all profiles.

Figure 3.1 provides an overview of the number of observed tracking requests and trackers (eTLD+1) for each page by profile. Generally, we see that *Firefox* profiles are tracked more than their *Chrome* counterparts. Furthermore, profiles in the US are tracked more than profiles from other regions. Finally, user interaction seems to have a significant effect in terms of tracking, while running browsers in headless mode makes only little difference. More details about our results are presented in the following sections.

Impact of Different Browsers (C5). First, we study the impact of the four browsers that we analyzed (i.e., *Firefox*, *Firefox headless*, *Chrome*, and *Chrome headless*) in terms of their impact on tracking. Regarding HTTP requests, we see that *Chrome*-based profiles make on average 2% (SD: 18.5%) more HTTP requests than *Firefox*-based profiles. Furthermore, we see that every 10th (SD: 1.5, min: 7, max: 12) HTTP request is a tracking request. We observed that for *Chrome*, every 10th HTTP request is a tracking request, while for *Firefox* it is every 9th. Overall, we identified for all Firefox profiles 12% more tracking requests than for the *Chrome* profiles. Only for four out of the 12 *Chrome* profiles, we could detect more trackers than for the respective *Firefox* profiles. However, on average, we identified 3.9 (SD:8.6) distinct trackers (eTLD+1) per page for the *Firefox* profiles and 3.9 (SD:8.1, min:

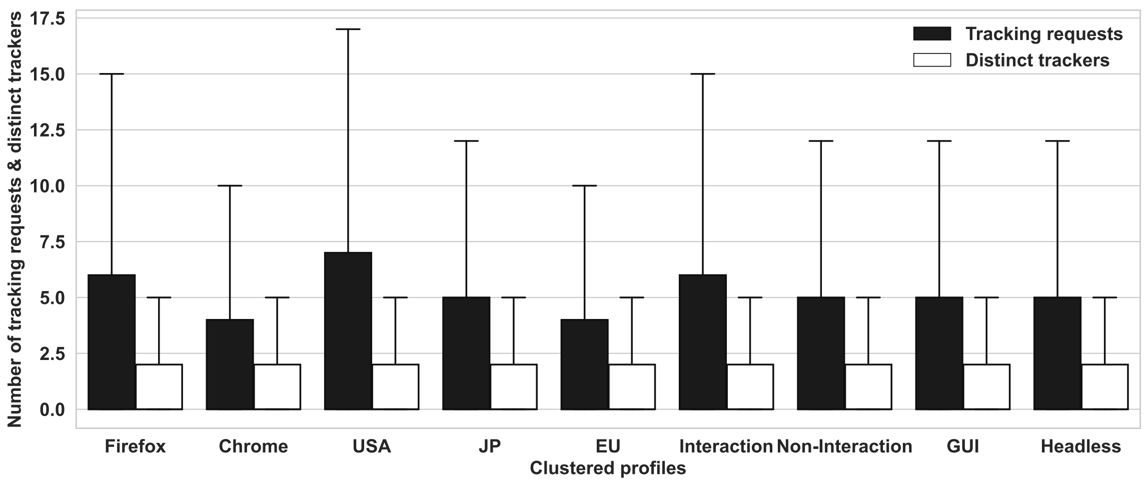

Figure 3.1.: Observed tracking requests and trackers by profile.

0, max: 68) distinct trackers for the *Chrome* profiles.Hence, the number of distinct tracking domains stays similar, while the volume of requests differs between the two browsers.

We turn to the effects of when a browser is used in headless or native ("GUI") mode. We observed only in two of the six *Chrome headless* profiles more trackers (10% per page) than in their counterparts. Overall, the headless *Chrome* profiles only contained 3% fewer trackers. When we run *Firefox* in headless mode, we noted almost reversed results. For four out of six *Firefox Headless* profiles, we could detect 5% more trackers than native *Firefox* profiles. Across all of these profiles, we see marginal differences regarding the number of trackers when we run browsers in headless mode. This is in contrast to previous work that has shown the importance of this feature [4]. However, in some of the profiles, we observe substantial differences, which indicates that the outcome of an experiment is not determined by the used browser mode exclusively. Moreover, different combinations of design choices mutually affect the results, highlighting the need for proper documentation.

Across all pages, the mean Jaccard similarity in observed distinct trackers for browsers *Chrome* and *Firefox* is 0.59 (SD: 0.32, min: 0, max: 1). Overall, we identified only 1% more trackers for *Firefox headless* profiles. However, we find a big difference in terms of identified trackers. Across all pages, the mean Jaccard similarity in observed distinct trackers for headless and non-headless profiles is 0.53 (SD: 0.48, min: 0, max: 1). Overall, the similarity in observed trackers comes with a medium Jaccard similarity for this category but with a significant standard deviation. While we observed a perfect similarity (1) for 19% of the pages, we found no similarity (0) for 11% of them (see also 3.2). This effect is magnified if we only look at the headless and non-headless browser where we find perfect similarity for 35% of the pages and no similarity for 34%. Hence, in the worst case, studies that only alter the browser (or the display mode) might find different results, depending on the analyzed pages.

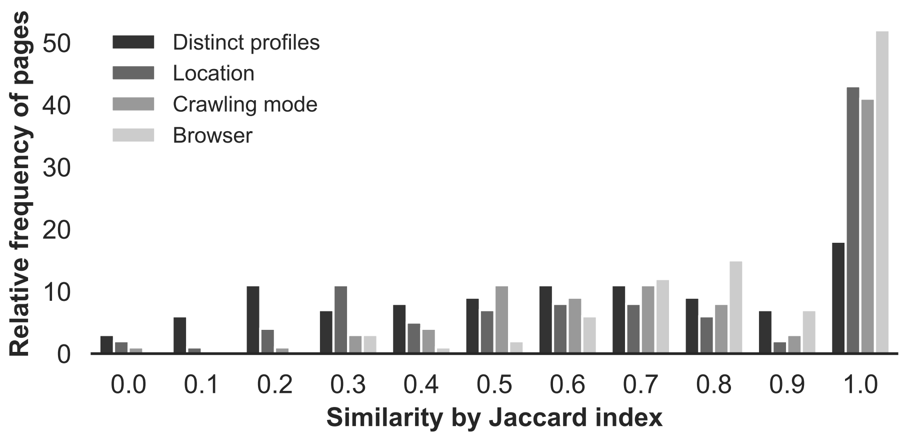

Figure 3.2.: Similarity of trackers on page level by profiles.

The distribution of the computed Jaccard values for each page is given in Figure 3.2 (black bar). Most pages (34.1%) always issue a very similar set of trackers no matter which profile visited the page (similarity $\geq$ 0.8). It is worth noting that we identified on such pages 1.9 distinct trackers on average. These pages only contain few trackers, but those are often present independently of the used profile. On 45.5% of the analyzed pages, we found a medium similarity (0.3 $\leq$ sim. $<$ 0.8) in the observed trackers. On those pages, we observed on average 5.2 trackers. Finally, 20.4% of the analyzed pages show almost no similarity ($<$ 0.3) in the observed trackers. On those pages, we observed on average 4.0 trackers. Thus, pages that include more trackers also include a different set of trackers based on the used profile. In the following sections, we discuss the impact of other criteria on the similarity in more detail.

Our results show that the number of tracking requests observed in the *Firefox*-based measurements is higher than in the *Chrome*-based ones. However, we did not find a statistically significant effect that running browsers in headless mode affects the number of observed trackers. However, we find a statistically significant difference (p-value < 0.001) in terms of identified distinct trackers.

We now describe our security analysis regarding CSP. Overall, we identified CSPs on 17.596 pages (10%). Compared to tracking analysis, we find very high similarity for CSPs. We find that on 16.355 pages (93%) the identified CSP headers are semantically identical. Hence, overall we get a Jaccard similarity of roughly .97. However, on the pages that served different CSPs (1.063), the mean Jaccard similarity is 0.68 (SD:0.25). Furthermore, we did not find that any profile had a significant impact on this phenomenon. This result is expectable since some of our features cannot be detected when the website is visited (e.g., user interaction) and, thus, cannot impact the results. Due to the low impact of our profiles on served CPSs, we dropped the CSP analysis in the other section because we found comparable results. Future work could analyze the impact of different browser profiles on more variable security features.

Impact of Simulated User Interaction (C10). Regarding simulating user interactions, our analysis shows that interaction on pages causes a sharp increase in HTTP traffic (on average by 20%) while the number of tracking requests increases by 35%. Hence, the amount of tracking requests increases disproportional with the number of all observed requests. For profiles with interaction, we observed on average 7.2 (SD: 8.8) distinct trackers (eTLD+1) per page and for the other profiles 6.7 (SD: 8.3). Thus, these high-level figures already indicate that the choice to simulate user interaction impacts the results of a study. When analyzing *Chrome* and *Firefox* separately, we see statistically significant (p-value < 0.001) differences. This again indicates that the effect of a single criterion cannot straightforwardly be attributed but that they jointly impact the results. For *Chrome* we find on average 6% (SD: 10%, min: -9%, max: 14%)) more HTTP requests when we perform interactions and, surprisingly, we see an average increase of 36% (SD: 6%, min: 29%, max: 43%) for the *Firefox* profiles. Of these requests we see that for *Chrome* 5.6% are tracking requests. For *Firefox* we see that 73% (SD: 21%, min: 43%, max: 92%) of these request are used to track users. This difference might be an artifact of our measurement framework and should be analyzed in future work in more detail.

Across all pages, the mean Jaccard similarity in observed distinct trackers for profiles with *interaction* and *non-interaction* is 0.67 (SD: 0.28 min: 0, max: 1). These results fit the observation that the number of observed trackers (eTLD+1) does not increase by a lot by user interaction. If the number of trackers stays similar, one can expect that the set of trackers per page stays similar. Almost half of all pages (47%) show a high similarity of more than 0.8 (see also Figure 3.2), which also indicates this trend. However, for a third (33%) of all pages, we find a similarity of 0.5 or less. This shows that while the overall similarity is quite substantial for a non-negligible number of pages, the results differ considerably.

Impact of Different Locations (C12). In this section, we want to analyze the regional effects of an experiment. On average, we see that profiles from the USA are tracked most in terms of distinct trackers (eTLD+1) on a page (6.93; SD: 8.5), followed by Japan (5.6; SD: 6.39), and EU profiles (4.49; SD: 5.42). These results propagate to the number of observed tracking requests.

Across all pages, the mean Jaccard similarity in observed distinct trackers for the profiles in the different regions is 0.62 (SD: 0.30 min: 0, max: 1). In terms of the analyzed criterion, the location has a significant effect on tracking and has only a limited effect on the difference in observed trackers. Half of all analyzed pages (50%) show a similarity of 0.7 or more (see Figure 3.2), and only 29% of the pages show a similarity of 0.4 or less—which should be accounted for in an experiment. This is in line with previous work that found that only a few online advertising companies altered their business model due to privacy regulations (e.g., withdrawing from the European market) [162]. Overall, we note that privacy measurements and analyses can vary up to 65% depending on the region (e.g., due to different legislation). Thus, our analysis finds that the region plays a crucial role in privacy measurements. These findings are consistent with previous work [162, 161].

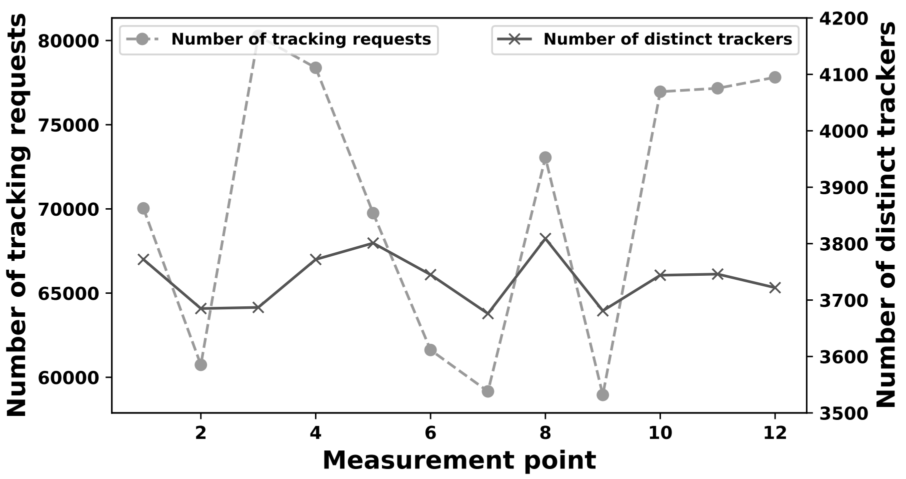

Figure 3.3.: Fluctuation in the number of tracking requests and domains for each day in the long running experiment.

Impact of Multiple Measurements (C4). Finally, we want to assess the temporal effects for measurement studies. In the following, we look at the absolute number of identified tracking requests and the number of observed distinct tracking domains (eTLD+1). Figure 3.3 shows the result of this analysis. Over twelve days, we saw a variation of up to 27% (max on day 3–80,274; min on day 9–58,951) in observed tracking requests. The standard deviation of such requests is 8,203. However, the number of distinct tracking domains remains almost stable during the experiment (variation of 3.5%). Our results suggest that depending on the day of each measurement, the number of tracking attempts—in terms of tracking request—frequently varies, but the companies (domains) that are active in the ecosystem remain stable. Thus, studies that analyze the ecosystem will find similar results, while studies that aim to analyze the extent of a tracking phenomenon might see different results based on the measurement day. In terms of replicability and reproducibility, this is challenging since even the same setup measures different levels of tracking on different days, which might lead to different conclusions of a study. Our results show how important repeated measurements are to draw more robust conclusions.

3.1.3. Limitations and Ethics

One limitation of our measurement is that—for scaling reasons—the site visits are not fully synchronized. We argue that this limitation will have minor influence on the results of our study, as the site visits still happen within a small time window (mean time difference is 3 min). We thus assume that sites will still hold similar content. The experiment design comes with the limitation that our crawler does not interact with websites as an actual human would, which is probably impossible in an automated fashion. From an ethical point of view, our crawler creates traffic on the visited websites that could be omitted and save resources, and we might see ads that might drain the budget of the advertising company. Since our crawler only visits each page once (once a day for the long-running experiment), we argue that these issues are minor and can be accepted.

3.1.4. Related Work

Recently different works, similar to our measurement study, focused on the comparability of various crawling tools used in Web measurement studies. Most recently, [87] compared how different measurement tools and setups affect the results of a study. Similarly to us, they used other locations and browser modes to perform the measurement. In their study, the authors focus on "request/traffic volumes", JavaScript libraries loaded, and known ad/tracking domains loaded. [4] presented a survey on tools used in Web measurements and performed an experiment to compare the outcomes of these tools. In their study, the authors compare metrics like request/response sizes or used cipher suits. Both works find, similar to our results, that different crawlers impact the results of an experiment. Cassel et al. found that mobile browsers receive fewer tracking-and-advertising requests than desktop browsers in a comparative study [25]. In contrast to our work, they focus on tracking and show differences between mobile and stationary devices. Similar to our approach to systematize and evaluate our community's researcher methods, other studies were performed by various authors in different domains [134, 6, 125, 9, 90]. Our work focuses on a different research object than the named studies, namely the Web.

3.2. On the Similarity of Web Measurements Under Different Experimental Setups

Modern websites are intricate and complex software applications, which offer a vast array of features. They often rely heavily on third-party services for their construction and operation. These services are embedded to dynamically load additional content, such as ads or fonts, which may only sometimes be under the control of the site operators [75, 97]. This dynamic loading process can introduce a non-deterministic set of objects on a page, potentially affecting commonly studied phenomena such as web tracking mechanisms [161] or HTTP headers [135]. Consequently, the same web page could present different objects during web measurement studies.

Researchers often resort to web measurements to comprehend various phenomena, like web tracking, security mechanisms, or the behavior of social media sites, that affect millions of users [162, 40, 83, 122, 1, 37, 30]. However, the dynamic nature of the web poses a significant challenge to these measurements. Tools such as *OpenWPM* [44] or custom-built crawlers are used to scale up these experiments. Despite this, the effects of different measurement setups and the root causes of measurement discrepancies still need to be more adequately understood. Prior research has indicated that even minor changes in a web measurement setup can significantly affect the results and conclusions of a study [4, 34, 87, 25, 135].

While previous studies have mostly explored the *effects* of such practices, they have yet to delve into *why* results differ. This study bridges that gap by investigating the effects of various measurement setups, providing a detailed illustration of differences in datasets resulting from the respective setups. We examine the similarity of embedded first- and third-party objects across five different measurement profiles. Leveraging these profiles, we conduct a large-scale web measurement covering nearly 25,000 sites and over 350,000 pages, forming the foundation of our analysis. Afterward, we construct *dependency trees* for each visited page and cross-compare these trees, enabling us to identify and quantify differences and determine to what extent they exist.

This approach aids us in fostering a deeper understanding of the comparability of privacy studies, by cross-comparing the similarity of different trees horizontally (i.e., nodes at a specific depth) and vertically (i.e., loading dependencies of an object). Our experiment contributes to establishing more robust measurement setups, ensuring reliable results and reproducibility for future work, and understanding why current measurements lack these aspects. Moreover, it provides insights into the comparability of different works.

In summary, our contributions include:

- **Differences in dependency trees.** We illustrate that trees obtained from different profiles present notable differences in dimensions, node types, and loading dependencies. These findings suggest that each page visit or measurement introduces a degree of variance, impacting the comparability and reproducibility of a study.
- **Causes of differences.** We identify the entity loading a node and the resource type of the node as primary factors influencing the differences observed in the trees. Specifically, we detail that a node's content type (e.g., iframes or images) and its loading context (e.g., third-party) are key drivers in introducing dissimilarities between web measurements.
- **Effects of measurement setups.** We examine the differences caused by minor changes in web measurement setups. Our findings reveal that even identical setups operating in parallel and visiting the same pages can yield significantly different results. Furthermore, we demonstrate that simple design choices (e.g., mimicked user interaction) can produce almost incomparable results.

3.2.1. Method on Measuring the Differences in Web Measurement Studies

In this Section, we describe our experimental setup (cf. Section 3.2.1.1) and our approach to measuring the differences between webpages (cf. Section 3.2.1.2).

3.2.1.1. Measurement Approach

This work investigates the causes of differences in the results of Web measurement studies when using different setups. We run semi-parallel measurements and define five profiles whose measurement results we compare. We develop our measurement framework along with recent findings on how to run robust Web measurements [180, 107, 161, 4, 179]. This design choice is intentional to allow the comparison of our work with previous work, which is currently lacking in our community [34]. Our measurement setup is based on the openly available framework of Demir et al. that implements a best-effort approach to conduct parallel Web measurements [34]. We describe the used framework in Appendix A.2.2 to foster reproducibility of our work.

In general, we use the following non-parametric tests: (1) the *Wilcoxon* signed-rank test to assess differences between two continuous variables, (2) the *Mann-Whitney U* test to determine differences between two independent variables, and (3) the *Kruskal-Wallis* test to assess if there are differences in the central tendency (median) of a continuous dependent variable across multiple groups. For all tests, we use a significance level of $\alpha = .05$.

Experimental Design. In the following, we describe the configuration of the five different browser profiles that we use in our experiment. We resort to these profiles since these types have been used by previous works [34] to highlight existing comparability issues in our community. Using methods or profiles that others have not used before could tamper with our intention to identify issues with existing approaches. All of these profiles are based on the *Firefox* browser, and we utilize *OpenWPM* (`v0.18.0`), a common and popular crawling framework [44], to capture the traffic we are interested in. We choose to utilize *OpenWPM* as it is widely used in the Web measurement community and, therefore, serves as a good foundation to conduct our experiments. As the framework already comes with a rich feature set, we only made adjustments to mimic user interaction, which is described later in this section. The named design choices regarding the measurement framework help to increase the comparability and reproducibility of our work.

We cross-compare these five profiles to understand the differences in the analyzed pages. The profiles differentiate in the used version, whether user interaction is mimicked ("user interaction") or not, and if the GUI of the browser is spawned or the headless mode is used ("GUI"). Two of the profiles (`#2` and `#3`) use the identical setup to directly compare the differences between two equal browser instances running in parallel, visiting the same pages. Table 3.2 lists the used profiles. We chose the "user interaction" and "GUI" configurations because, on the one hand, recent results show that such information is often omitted in setup descriptions [34] but could, thus, heavily impact comparability. Furthermore, both options are often omitted to speed up crawls to analyze more pages in an experiment. On the other hand, mimicked user interaction significantly impacts the embedded objects and third-party content (i.e., more content is loaded, for example, due to lazy loading). We chose the usage of a GUI as a parameter since previous works found a varying impact of this feature [34, 87]. In terms of the used browser version, we used

Table 3.2.: Overview of the used profiles. Profile #2 and #3 use the same setup.

#	Name	Version	Interaction	GUI	Country
1	Old	86.0.1	✓	✓	DE
2	Sim1	95.0	✓	✓	DE
3	Sim2	95.0	✓	✓	DE
4	NoAction	95.0	✗	✓	DE
5	Headless	95.0	✓	✗	DE

the most recent stable Firefox version available when we started the experiment (`v95.0`; release date 12/2021) and a version that is roughly one year older (`v86.0.1`; 02/2021). This distinction allows us to simulate differences one would face when comparing current results to ones from previous studies. Naturally, this approach will not reflect results that would have been obtained in previous works since websites, standards, and browser features change over time. Yet, our approach allows us to understand differences introduced by other browser versions. Furthermore, a roughly ten-month-old browser should still be supported by most websites. We make our browser profiles and the crawling technology openly available (cf. Appendix A.1).

To simulate a genuine user and the interactions of such a user is nearly impossible. However, such interaction can severely impact a site's behavior (e.g., lazy loading of content). In our profile, without user interaction (#4), we do not interact with the website, asides from waiting for it to finish loading or until the timeout of the framework is reached. All other profiles mimic simple user interaction with the visited page. Once the browser loads the page, we wait until the page finished loading (or a timeout is reached) and then simulate `Page Down`, `Tab`, and `End` keystrokes with short periods of delay in between. We resort to these keys as they will probably not result in loading a different page. The timeout and the specific keystrokes have been used by prior work (e.g., [37]). Such mimicked user interaction might also interfere with bot detection mechanisms in a way that such methods do not detect our crawler. We use the options of *OpenWPM* to define which user interface is used (options `native` and `headless`). To use an older browser version, we adjusted *OpenWPM*'s configuration to install the older binary.

Website Dataset. The basis of the set of websites to analyze is the widely used quasi-standard Tranco list [100]. We used a randomly sampled subset of sites from the list based on the pages' rank. We used the top 5k sites from the list and randomly selected 5k sites from each of the following buckets: 5,001–10k, 10,001–50k, 50,001–250k, and 250,001–500k. We used these 25k sites as a starting point to identify the pages to analyze. We visited the landing page of each of these sites three days before our experiment to collect 25 subpages (i.e., first-party links on the page) for each of them to get a broader view of a site's behavior [161, 34, 8]. We repeated the process recursively if the landing page did not hold enough links. We provide a list of all analyzed pages and sites in Appendix A.1.

3.2.1.2. Measuring Differences in Websites

A webpage can be modeled along the loading dependencies of the elements present on it (i.e., as a tree). We generally build the 'dependency trees' for each page to measure differences in websites based on the observed HTTP traffic (i.e., requests and responses). Each node in a tree represents an HTML element on a page (e.g., image, JavaScript, or CSS document), and the edges represent an HTTP request that leads to the content loading (child node). This approach is similar to methods used by previous works [161, 97, 75], which increases the comparability of our work. Naturally, the URL of the loaded resource is an excellent way to identify a node. However, we noticed similar or equal resources are often loaded via different URLs. One reason is that session identifiers or other ids assigned to a user (e.g., browser fingerprints) are included in requests as parameters. For example, the URLs `foo.com/scriptA.js?s_id=1234` and `foo.com/scriptA.js?s_id=abcd` could load the same or very similar script. Since we want to compare the trees, using the URL as an identifier and property to compare would distort the results because very similar or equal resources would not be compared. Comparing e.g., MD5 hashes of the loaded content is also unsuitable since sometimes the loaded content differs slightly as it includes the identifier. In our experiment, we still use the URL but adjust each URL to circumvent the mentioned challenges. To avoid complex (and maybe error-prone) content comparisons, we drop the values of query parameters and keep the remaining parts of the query string (e.g., `foo.com?s_id=`). Thus, depending on the browsers' profiles, different JavaScript libraries (e.g., different versions) might be loaded. However, in our analysis, we treat them as the same node. It is important to note that this step is performed during the analysis phase and *not* during the measurement. In our experiment, we had to apply this technique to 40% of the observed URLs across all profiles. We elaborate on the limitations of this approach in Section 3.2.4.

To build the trees, we resort to (1) JavaScript call stacks, (2) HTTP redirects and (3) (nested) iframe structures, which are all collected and provided by the measurement tool. Starting with the latter, *OpenWPM* stores the parent frame that issued a request, and thus we can assign each request to a parent frame and recursively build (sub)branches. We insert each frame at the corresponding position in a branch and combine branches if they all share the same parent node. Regarding JavaScript, we inspect the call stacks, which *OpenWPM* stores for each request. In each call stack, we inspect the latest entry (i.e., the event that issued the request). Thus, we identify the function and URL responsible for issuing a request and assign the caller as the parent node. We choose not to walk over the entire call stack because it does not directly indicate the dependencies of requests but instead of function calls. However, the latest entry always includes the URLs (request) responsible for the call. To find CSS dependencies, we also analyze the "call stack," which incorporates the CSS loading dependencies [19]. Thus, we handle them the same way as JavaScript dependencies. This is an artifact of the used *Firefox* and *OpenWPM* environment. All loaded resources that are *not* assigned to any branch are attached to the tree's root node (i.e., the loaded page itself). Eventually, each tree consists of all first- and third-party elements of a page, and each branch represents the dependencies that lead to embedding any given resource.

Identifying Tracking Requests. Our analysis aims to identify tracking requests as they are often analyzed in previous works [162, 40, 83, 122, 44]. Thus, analyzing such privacy-invasive requests allow one to put our results into perspective with other works. To identify the requests, we profit from the popular tracking filter list *EasyList* [43]). If an observed URL is on the list, we consider it a tracking request. We provide the used block list in the supplementary material of this work (cf. Appendix A.1) and discuss limitations in Section 3.2.4.

Comparing Request Trees. The core of our analysis is the cross-comparison of different trees generated when visiting the *same* page with the defined browser profiles. Our analysis only includes pages we crawled successfully with all five profiles. This approach ensures that we have enough data to compare page visits reasonably. All other pages are dropped from further analysis. This vetting results in rough dropping 34% of the pages in our experiment. This seemingly high number of dropped pages was not caused by a single profile but by combining all profiles. Each profile has a success rate of 89%, comparable to related approaches.

In essence, when comparing the trees, we analyze them along two different dimensions: (1) cross-comparing the parents of a node ('vertical tree analysis') and (2) analysis of siblings ('horizontal analysis of trees'). Starting with the latter, to understand the *horizontal* similarity of a node, we begin by computing the *Jaccard index* at a depth of one of each page's trees (i.e., the elements directly loaded by the page). Hence, we cross-compare which elements were loaded by all pages but exclude—at this stage—all objects subsequently loaded by such elements. After comparing depth one, we start a recursive approach for further elements. If we identified reoccurring objects in multiple trees with at least one child, we performed a similarity measurement of the children of these nodes. We repeat this step until we do not find any additional elements present in at least two profiles. We perform a bottom-up approach to assess the *vertical* similarity, starting with the last node in each branch. In this step, we only account for nodes at least at depth two, excluding nodes included by the visited page that did not load additional elements. We do so because the nodes at depth one always have the same parent, namely the visited page, and analyzing them is not interesting. On the one hand, we analyze the entire request chain of each child to understand deterministic of loading dependencies. To do so, we compare if two or more branches in the trees of interest are equal. This comparison allows us to understand how similar the additionally loaded resources of an embedded object are. On the other hand, we cross-compare the parent of a node in a branch to understand if the same resource always load specific content. This approach provides a context-specific perspective on which nodes are loaded and by whom. Appendix A.2.2 provides a visual overview of our comparison approach.

If not stated otherwise, we exclude in our analysis all nodes at depth one (i.e., the content loaded by the visited page) that cannot dynamically load additional content (e.g., plain text). We exclude these elements because they can only result in a "branch" with only one node and no children. Thus, if we included them, the reported numbers would not be sound because they would be biased. These "branches" would bias the analysis in that they show perfect similarity because they have no children and cannot load any. Therefore, our analysis would report that the branches of these elements are equal, which is true but would under-report dynamic effects on the web. Thus, removing them focuses the analysis on content that introduces dynamics into pages.

Computing Tree Similarities. To compute the similarity, we resort to the *Jaccard index*, that is used to gauge the similarity of sets and is defined as follows: $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$. By design, the index ranges from 0 to 1, while 1 denotes that the sets are equals and 0 that they have no element in common. We chose the Jaccard index because it allows us to compare and quantify the differences in included elements (based on their URL) on each page. To compare five sets, we computed the pairwise similarity between all sets and used the arithmetic mean value to state the similarity for a given page. To allow a straightforward interpretation of the compute scores, we use the following three categories: *high* (≥ 0.8), *medium* ($0.3 \leq$ sim. < 0.8), and *low* (< 0.3) similarity [34]. We choose not to compute similarities of entire trees (e.g., using the *Hamming distance* [179]) and resort to computing the similarity as it provides deeper insights into the changes in the relationships between the nodes. Using set operations and the Jaccard index, we focus our analysis on specific branches in multiple trees.

3.2.2. Results

Before detailing the results (Sections 3.2.2.1 to 3.2.2.4), we summarize the dataset in the following.

Success of Crawling Method. In our measurements, we successfully crawled 24,857 (99%) sites and identified 387k distinct pages on these sites. On average, we found 14.6 pages per site (min: 0; max: 25). Overall, our crawlers made roughly 1.66M page visits; we make the measured (raw) data openly available (see Appendix A.1). The sites our crawlers could not reach are not meant to be visited by a human (e.g., landing pages of content delivery networks or ad networks). On average, each profile visited 330k pages (SD: 25,263; max: 374,897; min: 312,941). Our analysis only considers pages successfully crawled by all five profiles, which applies to 17,851 (71%) sites, and 200,798 (55%) of the crawled pages and ensures that we have enough data to compare page visits reasonably. This reduction of sites and pages cannot be attributed to a single profile but is attributed to the combination of the profiles—each profile has a failure rate of <12% (mean: 11%).

Table 3.3.: High-level overview of the measured trees.

Tree	avg.	SD	min	max	Node(s)...	
nodes	84	99	1	12k	each present in X profiles (avg)	3.6
depth	3.6	2.2	0	30	present in all profiles	52%
breadth	44	58	1	12k	present in one profile	24%

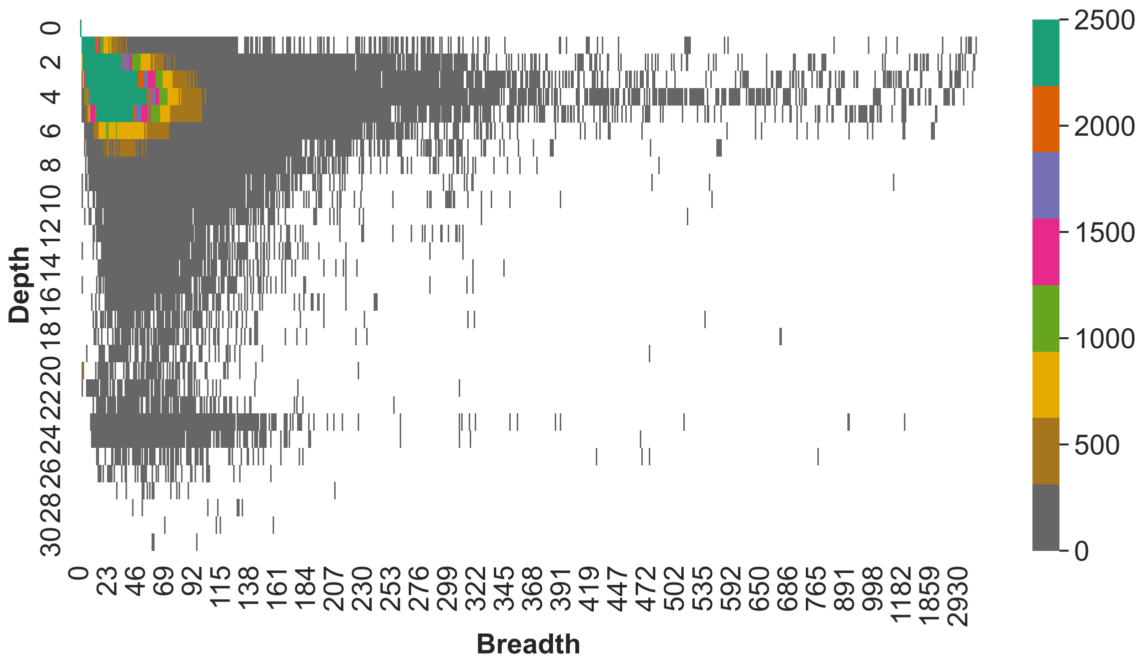

Figure 3.4.: Distribution of the observed trees' depth/breadth. We cap the scaling at 2500 to improve the readability.

General Structure, Size, and Differences of the Trees. The core of our analysis are the previously described dependency trees (cf. Section 3.2.1.2). In our experiment, we cross-compare five trees for the same page; 1.66M trees in total. Table 3.3 provides an overview of the measured trees, and Figure 3.4 shows the distribution of the depth and breadth of the observed trees. The high standard deviations (SD) of the trees' characteristics indicate that the structure of the trees varies. The distribution of the depth and breadth shows that relatively broad trees are often not very deep, while the wideness of a tree decreases the deeper they get. Nevertheless, more than half of the trees (56%) have a depth of less than six and breadth of less than 21. Concerning the general appearances of nodes in a tree that we observed in each of the five profiles (see Table 3.3), we see that each node appears on average in 3.6 profiles (SD: 1.7; max: 5; min: 1). This finding shows that deviations in the presence of nodes in the trees exist and are frequent. More precisely, roughly half of the nodes appear in all profiles, while a quarter of all nodes is present in only one profile. The results show that when one compares two different profiles 48% of the underlying data varies. On a high level, this indicates the severe impact of the Web's dynamic and used crawler on measurement studies that we aim to understand better in this work.

3.2.2.1. Differences in Node Dependencies

The presented figures suggest that snapshots of the same webpage taken almost simultaneously (in a best-effort attempt) show severe differences when one uses different browser configurations. In the following, we dive deeper into the differences we observed to understand the impact of the Web's dynamic on a practical level. Figure 3.5 shows the

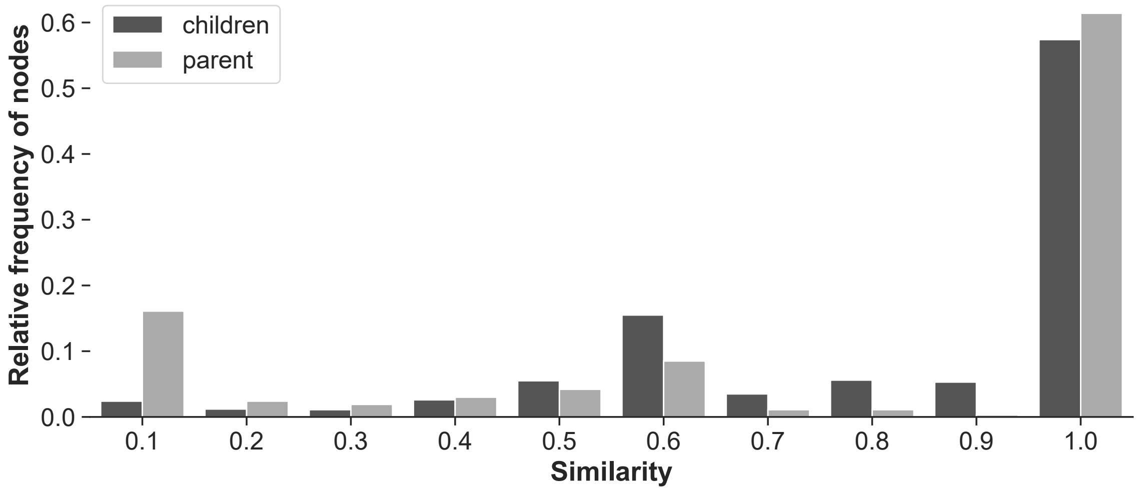

Figure 3.5.: Distribution of the similarities of all observed nodes per tree and the similarities of parents.

overall similarity of the nodes across all trees. We find that while roughly 60% of the nodes' children show high similarity, the remaining share shows a substantial fluctuation in the observed children. The similarity of the parents of a given node shows an almost perfect similarity for most nodes (61%). However, for numerous nodes, the similarity is low; 20% of the parents have a similarity of .3 or less. This observation shows us that while many nodes have a similar set of parents and children, differences in their relations can impact the measurement results' comparability. More precisely, the Web's dynamic makes it challenging to argue about loading dependencies (e.g., when analyzing ecosystems or mechanisms like cookie syncing).

Differences in Depth Levels. First, we look at the discrepancies of parties loaded on each depth, and cross-compare the nodes observed on each level individually (see Table 3.4). Thus, we compare depth one with depth one, depth two with depth two, and so on. Hence, we determine similarities between the nodes' depths and whether the nodes appear at different depths in other trees. This analysis allows us to understand *where* in a tree differences occur. Overall, we see a high similarity in the nodes' depths. The mean Jaccard index across all depths indicates high similarity, but we see a lower similarity if we exclude nodes at depth one without any children. This difference is expectable because several nodes at depth one reflect the content loaded by the visited page (e.g., texts or images), which is alike since we use similar clients in all profiles (e.g., no differentiation between content for mobile or desktop clients). Consequently, when we check the similarity for the nodes that appear in all trees, we see that they all appear in the same depth. Based on this analysis, we determine that if a node appears in all trees, it will appear in the same depth and that nodes directly loaded by the page show high similarity. Note that we are comparing *only* the depth of nodes, but not the nodes' loading chain. Hence, we might observe the same node at a given depth loaded by another parent. To dive deeper into the impact on the similarity of different nodes, we compare the similarity of depths of

Table 3.4.: Similarity of nodes at different depths.

Test	Value				
	cat.	sim.	SD	max	min
across all depths (all nodes)	high	.80	.21	1	.09
across all depths (only nodes with children)	med.	.74	.21	1	.09
nodes in all trees	high	.99	.21	1	.09
first party nodes	high	.88	.19	1	.09
third party nodes	med.	.76	.21	1	.09

first- and third-party nodes. Overall, the Jaccard index shows a medium similarity for third-party nodes and a high similarity for first-party nodes (cf. Table 3.4). On a high level, these results indicate that third-party nodes do not occur as stable as first-party nodes do. These figures indicate that studies focusing on third-party elements (e.g., cookies or trackers) will find varying results based on the setup. The reported numbers show differences based on the context but do not show their implications or the root of them.

Finally, Figure 3.6 shows at which depth different types of nodes occur in the trees. Visual inspection shows that most nodes occur at depth one (i.e., the elements embedded by the visited page) and that first-party objects are predominately embedded at depth one. Furthermore, nodes not used to track users (first- or third-party) are primarily present in the trees' upper levels (≤ 2). In contrast, almost half of the third-party nodes and tracking requests occur at deeper levels (> 2). Considering that the tree's similarity decreases at deeper levels, these numbers indicate that the comparability between experiments is not straightforward, especially for tracking requests and third-party nodes.

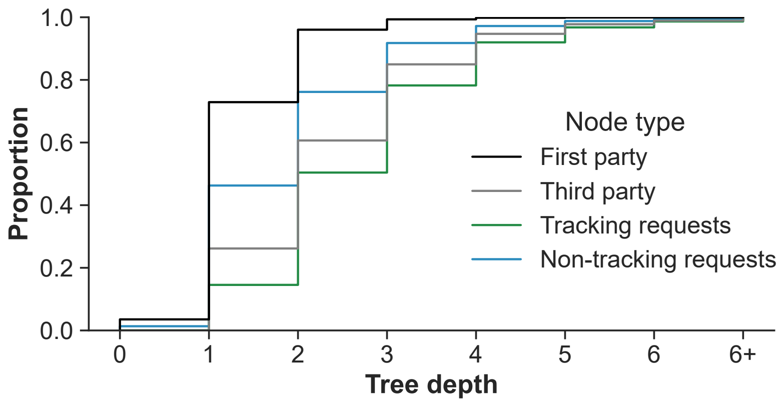

Figure 3.6.: Volume of different types of nodes in the trees. Nodes after depth six are combined into one.

Lessons learned

In summary, we discovered that even when analyzing pages almost in parallel using different profiles, the data observed in each profile differs considerably. We observed different nodes (HTML elements) in the measured tree representations of the same page. Finally, we showed that third-party components and nodes used to track users occur on deeper levels in the tree, which provide more variance. These findings indicate that a single measurement of a page will only capture a limited snapshot of the behavior of a page. Web measurement studies must account for this observation by understanding which elements of a page are dynamically loaded (e.g., due to properties of the setup) and which are stable.

3.2.2.2. Node Relations Across Different Trees

We have shown that differences in the observed trees exist, which opens the questions how they are introduced.

Cross-comparing Parents in Tree Branches. This part discusses the similarities of request chains to provide an understanding of the differences in the loading dependencies in the trees. We use the term *dependency chains* (or *request chain*) to describe all (grand)parents of each node. More technically, a request chain reflects the loading dependencies that initialized the loading of the analyzed node. For this analysis, we observe only nodes that appear in all trees to understand how the request chain of a node changes if it appears in another tree. We see that 75% of the nodes have the same dependency chains, which means that the same requests in identical sequences lead to the loading of the resource. In contrast, 18% of nodes (on average 12% per profile) have a unique request chain, which means that we observed this loading sequence in only one profile. If we exclude nodes on depth one, which have the visited page as a parent, we find that 57% of the nodes have the same request chains. Thus, on a high level, the results show that overlaps between the trees exist but that there is a considerable part that is varying.

The presented results show that loading dependencies are only sometimes stable. In the following, we shed light on *why* and *where* the request chains are changing. Furthermore, we aim to understand what causes such changes. We first focus on the nodes with the same dependency chains in all trees. Note that we exclude all nodes at depth one(42%) since their dependency chains consist of only one parent, which means they are naturally identical. We observed a long tail distribution in the numbers of these nodes per depth. On depth two, we found 21% of all nodes, on depth three 7%, on depth four 2%, and on all other depths combined 1%. Thus, most (94%) of the identical request chains are short (depth ≤ 3). These results propagate to identical chains we observed in only four or fewer trees. While these chains are short, they do not occur deterministically in all trees.

Table 3.5.: Effects of resource on loading dependencies.

(a) Same loading chains.

Node type	Same chains
main frames	90%
web sockets	88%
XMLHttpRequest	75%
JavaScripts	65%
style sheets	54%

(b) Lowest similarity.

Node type	Similarity
CSP reports	.10
Image	.25
Web socket	.27
sytle sheets	.31
web beacons	.34

As dependency chains tend to be non-deterministic it is interesting to analyze which kind of nodes—in terms of resource types—introduce variations. In the following, we test which resource types are always loaded by the same request chain at a level deeper than one. Table 3.5a provides an overview of the most common resource types that are always loaded by the same dependency chains. In contrast, Table 3.5b show the resource types that show the lowest similarity, which means that they are often loaded by varying dependency chains. The overlap between both tables (e.g., `web sockets` are present in both tables) is related to the fact that these types are often loaded by similar dependencies but that they are loaded in various contexts (e.g., libraries included by different scripts). On a high level, it is notable that the vast majority of first-party notes (86%) are loaded by the same request chain; in contrast only 56% of third party nodes are loaded via not changing dependency chains. These results replicate to nodes used to track users: only 28% of the tracking nodes are loaded by the same parents, but 66% of all non-tracking requests are loaded this way. The provided figures show that pages, visited with different profiles, include a considerable number of nodes in different ways (i.e., different dependency chains). This observation is particularly relevant for third-party nodes and nodes that are used to track users. For measurement studies, these findings implicate that high-level results, e.g., the presence of a specific node, can be compared, but the reasons why these results occurred (e.g., why a node is present) can differ noteworthy. Hence, studies regarding the ecosystem of a phenomenon can yield different results based on the used setup.

To better understand the variances in the dependency chains, we analyze which nodes are always loaded by the same node in more detail. Here, we limit our analysis to the nodes that appear at the same depth in all trees and to nodes that appear at least at depth two. The approach allows us to understand the dependencies of nodes and their parents. However, this filtering reduces the number of analyzed nodes to 7.5M (29%). Nodes can be triggered by different or multiple parents because, for example, different resources can load a JavaScript library. The results for this reduced dataset highlight that 61% of the nodes are triggered by the same parent in all five profiles. Thus, almost two-thirds of these nodes appear in all trees at the same depth loaded by the same parents, which means that analyzing them would provide the same results across all measurement profiles. Future work could use this finding to develop metrics that allow one to assess the expected 'measurement fluctuations' and to indicate the accuracy of an experiment. Furthermore, the results show that 63% of the parents show high similarity 17% show medium similarity, and 20% show low similarity. Our results demonstrate that the parent of a node and, thereby, the reason the resource is loaded differs in almost 40% of all cases. Thus, if one analyzes more than the presence of a resource, the results might differ based on the experimental setup (e.g., when analyzing ecosystems).

Now, we focus on the nodes with divergent parents to better understand the differences. More specifically, different parent nodes in other trees have triggered these requests (i.e., the dependency chains differ). The mean similarity of these nodes' parents is .33, which means that several resources (e.g., JavaScript libraries) are loaded by different parents. First-party nodes show a higher mean similarity in the sense of their parent (.54) than third-party nodes (.32).

Comparing the Children of a Node. Previously, we analyzed the parents of a node (vertical bottom-up approach). To understand how trees grow and which children and grandchildren a node loads, we reverse our analysis to all resources a node includes (horizontal approach). To do so, we test if (parent) nodes in the profiles load the same set of children. The results show that each node has, on average, 0.9 (SD: 6.4; min: 0; max: 4,613) children. Each visited page (i.e., depth zero) directly loads 31.7 nodes (SD: 36; min: 0; max: 4,613), on average. In contrast, most nodes on deeper levels (92%) only have one or no direct children (i.e., the nodes directly loaded by the node). The average number of children on all depths larger than zero ranges between 0.2 and 1. This distinct drop is expectable since many components cannot dynamically load additional objects. For example, an HTML image tag cannot load additional content aside from the image itself. This characteristic of the trees indicates that only some nodes are responsible for a tree's growth.

If we only look at nodes that have at least one child, we see a long tail distribution in the number of their children (depth 1: 3.5, depth 2: 2.7, depth 3: 2.0). Figure 3.7 displays the distribution in the number of children for each node based on their depth. Nodes on the upper levels of the trees have few or no children, while nodes deeper in the tree have more children. This observation seems counterintuitive, but most nodes cannot dynamically load additional content and, therefore, have no children. On deeper levels, the dynamically included objects load the content they want to insert into the page (e.g., ads). Since third-party and tracking components dominate the trees' lower levels, this finding indicates that such nodes could load substantially more content than their first-party counterparts. We dive deeper into differences between first- and third-party nodes in Section 3.2.2.3. However, outlier nodes with several children exist on all levels, which are responsible for the trees' growth. Thus, considerable parts of a tree are related to only a few nodes, which means that if they change, the tree itself changes substantially.

Previously we have shown that loading dependencies (e.g., parent nodes) are not stable, but we still need to determine if a node always loads the same set of children. To answer this question, we compare the similarity of the children of nodes by comparing the first-level children of nodes that appear in all trees (i.e., only directly loaded resources). Overall, we observed a medium similarity for a node's children (mean: .70; SD: .23; min: .09; max: 1). We find that the node similarity decreases with its depth if we analyze nodes with at least one child (at depth one). Overall, we observed a fluctuating but slightly decreasing trend

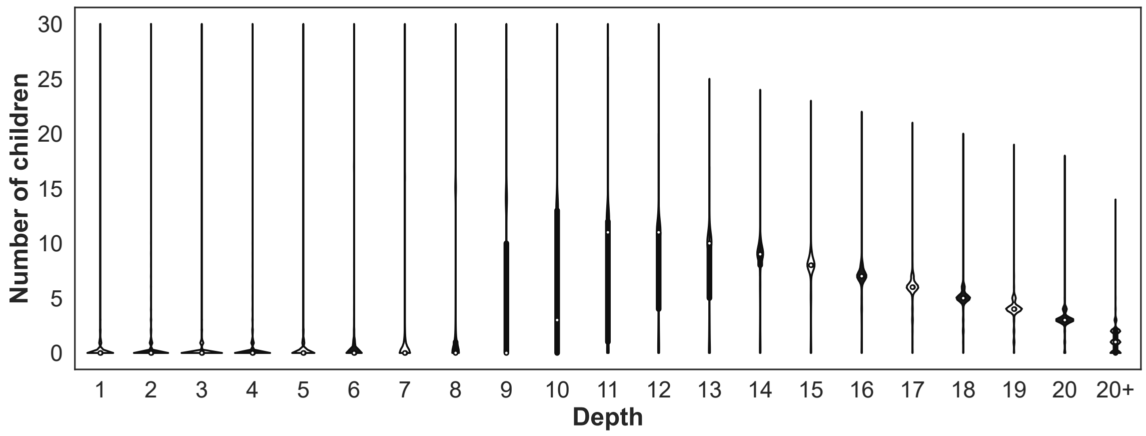

Figure 3.7.: Number of children each node has at a specific depth. We capped the number of children at 30 and combined all depth level deeper than 20 into one.

.

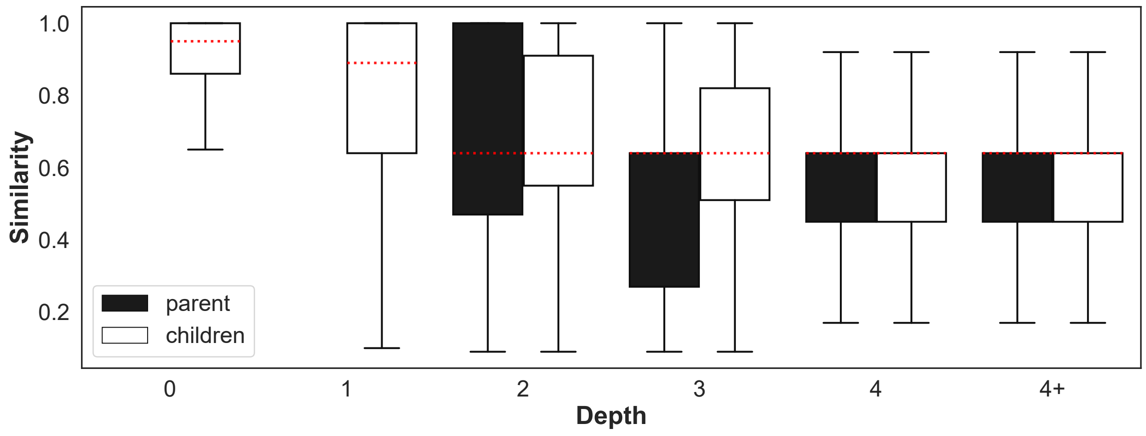

Figure 3.8.: The similarity of children and parents. We combined all nodes occurring at a depth deeper than four ("4+") to increase readability.

in the similarities, starting to increase again in deep branches. This observation roots in the fact that on deeper levels, only a few nodes exist ($n < 100$), which are loaded by a small set of nodes. Figure 3.8 shows the decreasing similarity based on the depth, and Figure A.3 in Appendix A.2.2.3 provides an overview of the children and parents' similarity for different resource types per depth. One result from this observation is that a phenomenon of interest at deeper levels, it is possible that other measurements will not find the same. However, if a node is present at higher levels in a tree, it is also more likely to be present in other measurements. The Wilcoxon signed-rank test found statistical significance between the number of children and their similarity (p-value < 0.001), that is, nodes that have many children often load different children.

Understanding Implications of Resource Types. To gain a deeper understanding of a node's type on the dissimilarities of a page, we examine the impact of its resource type on

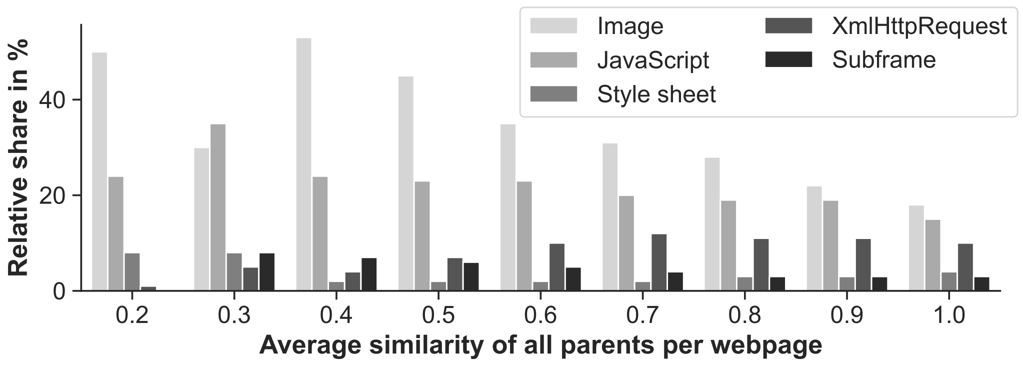

Figure 3.9.: Resource types by average *parent* similarity.

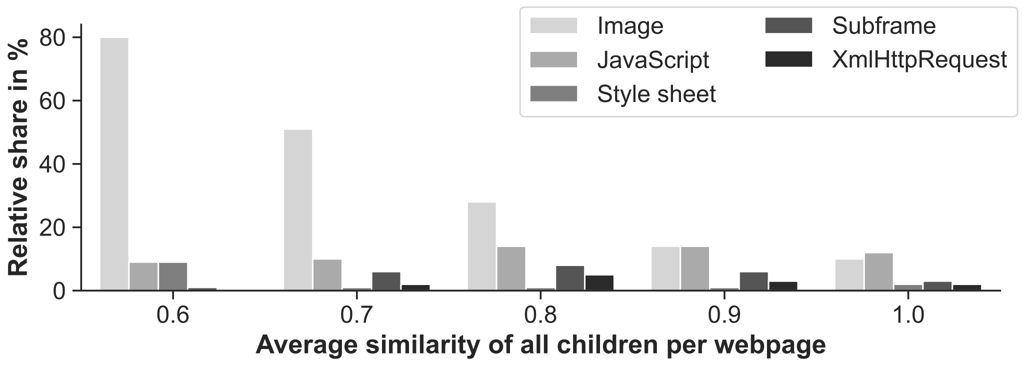

Figure 3.10.: Resource types by average *child* similarity.

the similarity of its children and parents. Figure 3.9 provides an overview of the distribution of resource types and the average similarity of parent nodes concerning the overall similarity of a page; specifically, it shows the share of content of pages with comparable similarities. One can see that on webpages with low parent similarity, the node types `image`, `script`, and `subframe` (e.g., iframes) show the highest relative share, which indicates that they are mainly responsible for dissimilarities. In contrast, Figure 3.10 provides an overview of children's similarities. Again, images show the highest similarity for child nodes, while JavaScript nodes have the highest share. The Kruskal-Wallis test found a statistical significance that the resource type of node affects the similarity of children and parents of a node. On a high level, the results indicate that specific types of nodes (e.g., images) cause more dissimilarities than others (i.e., `XMLHttpRequest`). However, a deeper analysis of the resource types shows that subframes have the most significant impact on the similarity of the trees. Pages *without* any subframe show high average similarity (parent: .86; children: .90) while pages *with* subframes show a medium average similarity (parent: .72; children: .77). These results provide different challenges since iframes (subframes) are widely used (e.g., to embed ads), and their variance introduction in the results tampers with the comparability, reproducibility, or replicability of a Web measurement experiment.

Lessons learned

This section shows that the dependencies between the nodes (i.e., loading dependencies) are not stable across the measurement profiles. This finding implies that a single measurement snapshot of a page only holds one of the many ways a page can embed an object. Furthermore, we have shown that the node's resource type influences the children loaded by a specific node, as certain types (e.g., JavaScript) tend to load a varying set of children. Hence, when designing a Web measurement study, it is essential to understand (a priori) how a page could include the entities of interest into a page and how the entity type could affect a study.

3.2.2.3. First- and Third-Party Context

We have shown that the loading party affects the similarity of the children, and the results indicate that third-party resources are less deterministic. This section provides a deeper understanding of this observation.

Understanding Implications on First-party Level. We start by analyzing the nodes loaded in the first-party context and aim to understand which changes they cause in the trees. Overall, only 32% of nodes are loaded in a first-party context. However, first-party objects are primarily present at depth one and two, and (only) in these levels they dominate third-party nodes (depth 0: 99%; depth 1: 55%). To understand how consistent a first-party node appears on a page, we test we examine the frequency of such nodes appearing in other profiles. Our analysis shows that, on average, the nodes at depth one appear in 4.5 of 5 profiles. This observation shows that most pages load a nearly identical set of first-party resources, regardless of the measurement setup. On deeper levels (>1), we see a similar picture with a minimum of 3.6 of the profiles containing the same first-party nodes (max: 4.8). Hence, comparing resources loaded in the first-party context is robust for similarly configured crawlers (i.e., same browsers with similar interaction profiles).

While the presence of first-party nodes is comparable, assessing if they also load a stable set of children is essential. If we look at the similarity of first-party nodes' children, we find a high similarity for these nodes (mean: .86; SD: .20; max: 1; min: .09). Thus, we see that, on the one hand, pages load a very similar set of first-party nodes across different visits and that, on the other hand, these nodes load a similar set of children. These observations are as expected since the website providers control these resources and selectively deliver different content based on, e.g., the used browser. Thus, measurements focusing on first-party components are expected to be stable and produce comparable results, even if the measurement setup changes to some extend. Note that some changes, in our case user interaction, can have far reaching impact on the results and, therefore, tamper with the comparability of the results. These numbers validate that our framework, measurement approach, and analysis method are valid since they produce expected results.

Understanding Implications on Third-party Level. While first-party nodes and their children are quite stable in their embedding, third-party nodes show probably a different behaviour due to their high dynamism [161]. In the following, we analyze the effects of third-party nodes in the measure trees. In our measurement, 68% of all nodes are loaded in a third-party context and they belong to 21,154 distinct third-party domains. Starting at depth three, third-party nodes dominate first-party nodes (on average 95%). Thus, the third-party nodes caused the vertical growth of the observed trees.

Similar to the analysis of first-party nodes, we analyze the appearance of third-party nodes to understand if they occur in all profiles at similar positions. More specifically, we test if an observed third-party node appears in all profiles and how their appearance frequency changes. Compared to first-party nodes, third-party nodes appear less stable across the different profiles: our analysis shows that the third-party nodes at depth one appear in 3.9 profiles, on average. However, at deeper levels (>2) we observe a sharp decrease (mean: 3.3, SD: 1.6, max: 5, min: 1). These results show that the immediate inclusion of a third party (i.e., at depth one) is similar across all profiles, while the subsequently loaded third parties are less stable.

Finally, we analyze the similarity of third-party nodes' children. The Jaccard index shows an average medium similarity of .68 (SD: .23; max: 1; min: .09), and across all nodes we identify that, generally speaking, third-party nodes have much more children (increase of 84%) and trigger more HTTP requests (increase of 150%) than first-party nodes. Compared to first-party elements, our results suggest that a exhaustive experimental setup is needed if a study focuses on a third-party phenomenon. As a result, the reproducibility, replicability, and even repeatability of studies suffer. This finding is independent of the depth of the nodes.

Lessons learned

This section showed distinct differences between nodes loaded in the the third- and first-party context. The similarity of nodes in the first-party context is high; however, we observe lower similarity values for third-party elements such as trackers or ads. This observation primarily impacts privacy-related studies investigating such content (cf. Sections 3.2.2.4 and 3.2.3.3). Therefore, future studies focusing on third-party content should handle such dynamics to ensure their results' generalizability, completeness, and comparability. One way to do so is to perform multiple measurements—using different profiles—of the same page to capture a complete view of its behavior.

3.2.2.4. Assessing Setup Implications

Previously, we have shown that the analyzed sites introduce notable variance in the performed measurements. While this is challenging itself, it is essential to understand

Table 3.6.: Implications depending on different profiles.

#	Name	Nodes	Third party	Tracker	Depth	Breadth
1	Old	19.62M	13.42M	3.32M	28	11 649
2	Sim1	19.41M	13.24M	3.21M	30	4562
3	Sim2	19.34M	13.19M	3.20M	29	4258
4	NoAction	14.53M	9.25M	1.91M	30	4953
5	Headless	19.39M	13.22M	3.20M	30	4562

which impact the measurement setup on the outcome of an experiment has. Table 3.6 provides a high-level overview of the observed trees for each used measurement profile. In terms of dimension (e.g., number of nodes or depth), most of the trees are of similar size, but some characteristics differ (e.g., the max. breadth of a tree in profile #4).

Comparing Profiles with the Same Configuration. To understand the impact of the Web's dynamic on the comparability of different studies we compare two profiles that use the same configuration (#2: `Sim1` and #3: `Sim2`; cf. Section 3.2). Overall, the trees of both profiles have similar dimensions in terms of: (1) first-party nodes, (2) third-party nodes, (3) depth, and (4) breadth of trees. These figures indicate that both profiles crawled the pages of interest at a similar success rate; therefore, comparing them is valid. Concerning the similarity of the trees, the results show that on the upper levels (≤ 5) the trees are highly similar (mean: .92), but that the similarity decrease on deeper level (mean: .75).

Table 3.7 provides an overview of the observed differences between all profiles. The results show a distinct difference in the relative number of perfectly similar nodes (i.e., nodes that appear in the same depth in both profiles). It is important to note that even if the numbers across all profiles are similar (expect for profile `NoAction`) the impact on the results are different. It is also not the case that the same node across all measurements show a perfect similarity (see the previous sections). The results show that first- and third-party nodes show notable differences in the set of loaded children (i.e., they differ in 18% of the cases) in both profiles.

Browser with an Outdated Version. To assess how comparable two measurements are that analyze the same websites but with different browser versions, we compare the measurement using an outdated browser (profile `Old`; #1) with profile `Sim1` (#2). The results are similar to the comparison to profile `Sim2` (cf. Table 3.7). Thus, using an outdated browser version has similar effects on the comparability of an experiment than using the same setup. We expected such results because we visited the same pages, presumably supporting the outdated version; therefore, the results are similar to the profile that uses the same configuration.

Mimicking User Interactions. To understand the impact of simulated user interaction, we compare profile `Sim1` (#2) with the profile that does *not* mimick user interactions (profile #4;

Table 3.7.: Profile differences compared to profile #2 (`Sim1`).
❷: Starting at depth two. ❶: For nodes with at least one child.

	`Sim2`	**`Old`**	**`NoAction`**	**`Headless`**
		First Party nodes' children		
perfect similarity	82%	80%	67%	82%
no similarity	4%	5%	8%	4%
		Third Party nodes' children		
perfect similarity	75%	73%	64%	75%
no similarity	13%	15%	22%	13%
		First Party nodes' parent		
perfect similarity	94%	93%	92%	94%
no similarity	6%	7%	7%	6%
		Third Party nodes' parent		
perfect similarity	65%	63%	64%	65%
no similarity	30%	31%	31%	30%
		Dependencies		
parent similarity (mean) ❷	.71	.70	.70	.71
child similarity (mean) ❶	.83	.84	.74	.84

`NoAction`). Similar to previous works [34, 161], we find that simulating user interactions cause much more HTTP traffic, which leads to larger trees. Comparing the number of nodes, we see that profile `Sim1` has 34% more nodes than profile `NoAction`. Furthermore, we find that profile `Sim1` has more third-party nodes (36%) than profile `NoAction`, and that each node has *less* children (15%). The Mann-Whitney U test found statistical significance of the mimicking of user interactions on the nodes' depth level (p-value < 0.001). Hence, the profiles with user interaction have more nodes at a deeper level. Both results show that interactions cause a vertical growth of the trees, and new nodes load fewer children than the nodes before mimicking user interaction. More importantly, our results indicate that mimicking user interaction changes the result when measuring a page.

Compared to all other profiles (see Table 3.7), profile `NoAction` show the highest variation (i.e., fewer perfectly similar nodes and more nodes with no similarity) across metrics except first party nodes. This observation shows that the introduction of even simple simulated user interactions can make a measurement more stable.

Using Headless Mode. To understand the impact of utilizing the headless browsing mode, we compared the profile that uses this mode (profile #5; `Headless`) with the profile `Sim1`. The results (cf. Table 3.7) indicate that both profiles introduce a similar variation to the results. Thus, we could not determine a positive or negative effect of utilizing a crawler in headless mode. Note that the results of both profiles cannot be straightforwardly compared as they both introduce a notable variance into a measurement; however, the magnitude of the variance is similar. This finding aligns with previous work that found no statistically significant impact of using the headless mode [34], in contrast to (older) works that found such impact [1].

Lessons learned

The results presented in this section indicate that the outcome of a measurement can differ significantly depending on the utilized browser configuration. Even using the same configuration and visiting the same page simultaneously, the nodes and dependencies of the elements on a page can differ. Thus, it is crucial to be cautious when making conclusions based on a small sample set since a significant part of the Web is too dynamic. To overcome this challenge, a measurement should carefully select the used configuration(s). Finally, developing a metric to understand a measurement's potential error/variance is vital to gauge the precision of a web measurement study.

3.2.2.5. Understanding Implications on DNS Level

In this section, we analyze whether different DNS resolutions affect the measured trees. All used VMs run in the same network segment and use the same DNS server for name resolution. Hence, mechanisms like load balancing could most likely have the highest impact on the name resolution.

Associated IP Addresses. Previous works have shown that based on the location or due to load balancing mechanisms, servers might deliver different content for the same page (e.g., other HTTP headers [135]) to the client. Therefore, we analyze which IP addresses are used by the clients when they access a webpage. We first compare the similarity of IP addresses associated with the DNS record. More specifically, we compare the retrieved available IP addresses from the DNS servers to access the corresponding host. For this analysis, we only focus on the nodes that appear in all trees (52% of all nodes). Overall, we see that 69% of all these nodes have the same DNS response (similarity = 1) in the sense of associated IP addresses. The mean similarity of these nodes' IP addresses is .82 (SD: .31; min: 0; max: 1). When we test the IP addresses that are not the same for the nodes, we see that 57% of DNS requests are resolved over CDN providers (cf. Section 3.2.1.2). Since CDNs are known for load balancing of Internet traffic, it is expected to retrieve different IP addresses for the same host. Yet, Roth et al. have shown that such might have a negative impact on security mechanisms as different HTTP headers might be delivered [135]. However, it's worth noting that we see records that contained *IPv6* addresses (7% of nodes) while it was not the case in other profiles also for CDN providers. Regarding the use of IPv6 addresses by a client, we find that the number of such records with an IPv6 address differs substantially across the profiles (i.e., Profile #2: 4.6M, Profile #3: 6M records). It is also notable that we have identified 2.7M (20%) DNS responses with an IPv6 address, but at the same time, at least one profile for the same node did not include an IPv6 address. Thus, our results show that the results of studies on IPv6 may differ depending on the analyzed metric due to this dynamic. In Appendix 3.2.2.5, we discuss the IPs used by the client.

Impact of DNS over HTTPS. In this paragraph, we analyze whether *DNS over HTTPS* (DoH) functionality influences the result of different measurement setups. DoH is a relatively new technique that the *Firefox* browser offers to resolve domain names. Hence, it could similarly affect the outcomes of a measurement study as 'traditional' domain name resolution. In this analysis, we are only analyzing the nodes present in all trees. In general, we see that DoH was used for 12% of the request for the profile without user interaction (#3; `Sim2`), while the profile that runs the outdated browser (#1; `Old`) has no records for DoH. This is because the used browser version does not support DoH in its standard configuration. For the other profiles (#2, #4, and #5), we see that DoH was used for 24% of the requests. Thus, we identify that simple user interaction during crawling directly implicates the DNS level results. The results show that DoH seems to be influenced by the profile used, which implies that different IP addresses could be used for the same page in a measurement study. Potential effects of this have been shown by Roth et al. who showed that websites could deliver different HTTP headers (i.e., CSPs) based on resolved IP address [135]. These findings suggest bidirectional implications: (1) the selected profile type can cause different DNS results, and (2) the DNS resolutions impact the results of a measurement study. Similar to our other findings, we observed that DoH seems to be implicated by the used profile.

Used IP Addresses by Client. In this section, we analyze the selected IP addresses from the DNS response to understand whether the profiles accessed the same servers. In our experiment, only 67% of the analyzed pages were accessed using the same IP address in all profiles. We see that the mean similarity of selected IP addresses is .78 (SD: .34; min: 0; max: 1). Compared to the described results on the similarity of the nodes per depth, we identify that the similarity of the used IP addresses increases for deeper nodes in the trees (e.g., depth 0: .73, depth 1: .83, depth 6: .65, depth 12: .90 , and depth 24: .99). When we check effects' of third party we see first party nodes show more similarity (7% of nodes). The results also show that 75% of nodes that were accessed by at least one profile with a different IP address belong to CDN providers. The one-way ANOVA test found a statistical significance between the mean similarity of children and parents and the similarity of selected IP addresses by the client (p-value < 0.001). This finding implicates that if the browsers use different IP addresses to load a resource, the similarity of the children changes.

Figure 3.11 provides an overview of the similarities that we observed concerning the associated and used IP addresses by the different profiles. The *associated* IPs are all IPs returned by the DNS query, and *used* IP is the IP chosen by the client. As one can see, on the first depth levels, the similarity is high and often almost perfect (the median is one), while it decreases for nodes in the middle of the trees. Thus, the results suggest that dynamically loaded (third-party) content is often loaded from different IP addresses, which could lead to artifacts that different content is loaded.

Lessons learned

In summary, our results show that if multiple measurements are performed from the same location, using the same DNS server on the system level, different resolution results can occur, which might mean that different servers are used to visit a page (e.g., due to load balancing or other network-level aspects). Even if this finding does not have to mean that the loaded content is different, previous work has shown that it can [135]. Hence, dissimilarities can occur not only due to dynamic content on a page, but also due to dynamics on other layers. Thus, one needs to carefully consider *where* and *why* differences may appear to circumvent them.

3.2.3. Case Studies

In the following, we present three case studies to put our results presented in the previous section in a practical context.

3.2.3.1. Unique Nodes

When analyzing novel phenomena or emerging technologies, Web measurement studies often need to find the 'needle in the haystack.' We analyze unique nodes to understand the chances of finding 'the needle.' We define nodes as unique if they appear in *only* one tree. More specifically, we consider a node unique if and only if it appears in only one tree, ignoring the depth (i.e., the URL corresponding to this node is only present once in our dataset). Overall, we see that 6M of the nodes (24%) are unique. Our observation shows that 37% of such nodes belong to tracking requests and 90% to third parties. Thus, using different crawlers might yield varying results, especially when analyzing new tracking techniques.

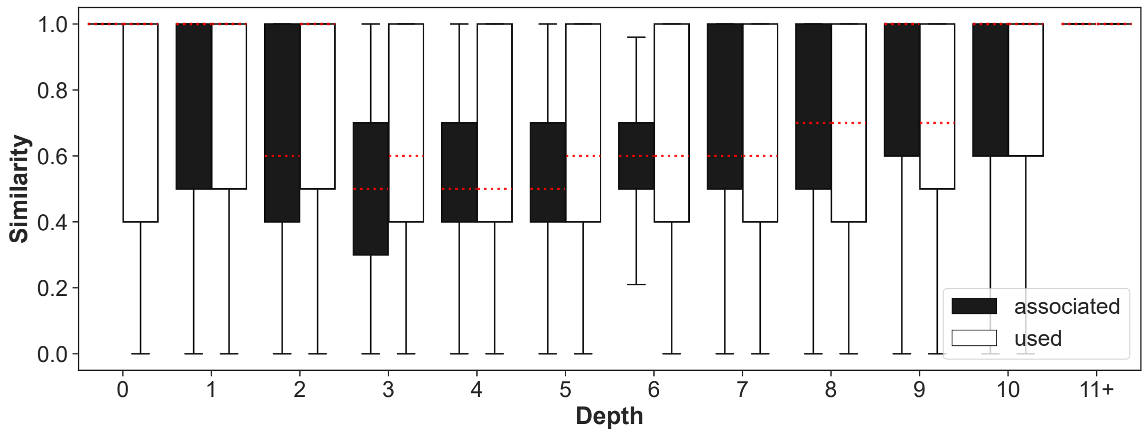

Figure 3.11.: Similarity of associated and used IP addresses by used profiles. To increase readability, we combine all nodes after depth eleven ("11+").

On average, unique nodes appear at depth 2.7 (SD: 1.9) and 22 % of these nodes are at depth 1. All of these nodes are capable of dynamically loading content based on their resource type: iframes (17%), JavaScript (15%), or XMLHttpRequests (13%). Regarding the site that delivers the resources, we see that common ad networks are the top hosters of such content (e.g., *googlesyndication.com* (20%)). While these eTLD+1s are pretty popular, they host advertisements that appeared uniquely in our experiment. These results are in line with the results regarding tracking requests. On average, 6 % of all nodes in a tree are unique. The results show that the relative share of such nodes increases with the number of total nodes.

3.2.3.2. Implications on Cookies

In the following, we analyze the impact of our findings on the setting of cookies. As per RFC 6265, we uniquely identify cookies by name, path, and domain [14]. We analyzed cookies as they are widely studied (e.g., [162, 30, 45, 24]). Overall, we observed 2.2M cookies on the analyzed sites and each profile set 438k (SD: 39k; min: 370k; max: 459k) cookies, on average. The profile without mimicked user interaction (#4; `NoAction`) has the fewest cookies (370k), and the other profiles have a similar number of cookies (mean: 455k). Since we want to understand if different profiles can yield different results, we test how the appearance and similarity of the cookies differ when we visit a webpage. We see that only 32% of the cookies appear in all profiles and 42% only in one profile. To better understand this observation, we cross-compare the similarity of the observed cookies per webpage. The mean Jaccard index across all cookies indicates a medium similarity of .70 (SD: .27; min: .0; max: 1;). When we compare profiles with interaction with the profile `NoAction`, we see less similarity, on average .59 (SD: .44; min: .0; max: 1). The results show that for 440 (0.2%) distinct cookies, at least one of the security attributes (e.g., `same site`, `http only`, or `secure`) has been set differently. Our analysis highlights that even if the same webpage is analyzed, the observed cookies provide substantial variance, which means that comparing measurements is not straightforward. This is surprising because these are hard-coded cookie attributes that one would not expect to differ.

3.2.3.3. Tracking Requests

Finally, we analyze the implications of our results for tracking requests, a widespread and often studied phenomenon on the Web. Overall, we see that 22% of the nodes are used for tracking purposes, and we observed a mean similarity of .53 (SD: .27; min: .09; max: 1;) for these nodes. The mean Jaccard index for the similarity of tracking nodes' children across the observed trees is .62 (SD: .21; min: .09; max: 1;), and for non-tracking nodes .75 (SD: .23; min: .09; max: 1;). It is worth noting that tracking nodes have fewer children (mean: 1.7) than non-tracking nodes (mean: 3.7). Thus, trackers are less stable in embedded children than other nodes, making them more challenging to analyze. Testing the similarity of parents of tracking nodes shows a lower similarity for the tracking nodes (.53; SD: .27;

min: .09; max: 1;). This finding indicates that tracking requests are triggered by much more different requests than non-tracking requests.

We analyzed the distribution of tracking nodes in our trees, and we see they mostly appear in the upper parts. Of the observed tracking nodes 9% appear at depth one, 32% at depth two, 36% at depth three, and the remaining 24% on the deeper levels. Embedded trackers significantly impact a tree; slight changes in the upper level can cause different trees, as they are primarily found in the upper parts and have a lower similarity. When looking at the parents of the tracking nodes to understand who is responsible for loading such nodes, we find that tracking requests are often triggered by other trackers (65%), which are primarily loaded in a third-party context (82%). Overall, we see that, on average, 58% (SD: 43; max: 100; min: 0) of the tracking requests were triggered by first-party requests and 42% of the tracking requests were loaded by third-parties. Our analysis of the parent of the tracking nodes shows that 46% of tracking nodes are triggered by JavaScript resources, 34% by subframes, and 15% directly by mainframes.

3.2.4. Limitations and Ethics

Our method has limitations and different ethical implications discussed in the following. Aside from artificially mimicked user interaction, we do not interact with the visited pages. Our approach does not observe content that is only displayed based on user action or content only loaded in a particular use case (e.g., triggering of a payment service). Thus, content that is only displayed based on user action (e.g., after interaction with a "consent notice"), a specific state of the user's browser (e.g., logged-in users), or content only loaded in a particular use case is not observed by our approach. We must highlight that our crawler does not interact with consent notices. However, since all measurements are performed from the same location, these effects affect all profiles similarly, and comparing page visits is still suitable. This circumstance means that our results can be seen as a lower bound but also that they are not complete. However, it must be noted that implementing a system that automatically generically interacts with any given page to trigger a different state is out of the scope of this work. Our approach follows best practices and uses well-established tools to conduct Web experiments to reduce the effect of the mentioned limitations.

Furthermore, our method of combining requests from the same origin, based on their path, can lead to merging branches that do not originally belong to each other. However, using all URLs observed in the trees will (unrealistically) increase the observed differences due to, e.g., session identifiers in the URL. Furthermore, our approach has the limitation that different URLs might look the same after purging the parameter, even if they load different content. Accordingly, when we build the trees using these URLs, branches might be collapsed into one because the identifier (parent node) might no longer be clear. Overall, our approach will lead to smaller trees and will more likely underreport the scale of the problem (lower bound) than overreport it.

Our analysis uses the popular tracking filter list *EasyList*. The list is a crowd-sourced attempt to identify web tracking, among other things.and, therefore, might be incomplete or, to some extent, wrong. However, we assume that such errors only have a marginal impact on our results. Finally, *EasyList* is only one of many blocking lists. Combining multiple lists could increase the comprehensiveness of detecting trackers, which we assume would not significantly change our findings or takeaways. Adding blocklists could also result in a more distorted measurement because lists like *EasyPrivacy* do not focus on tracker blocking [32].

In the following, we discuss the limitations of this study. Our study does not include or directly affect any human subjects. However, our large-scale measurement has ethical implications that must be discussed. Our experiment artificially generates website traffic that would not emerge without our experiment. This traffic will use resources (e.g., energy or bandwidth) that would otherwise be unused and could result in additional costs for the service providers. Since our traffic is distributed over several thousands of sites and the traffic towards individual sites is limited to the loading of up to 25 pages, we argue that the traffic generation is reasonable for our experiment. Another aspect to discuss is that if websites serve ads to our crawler, this could burn the ad budgets of advertisers. However, due to the finite number of measurement runs, we argue that these costs are negligible. These discusses aspects apply to all Web measurement studies and our used practices are considered state-of-the-art and are accepted by our research community [34, 87, 25, 135, 180, 107, 161].

3.2.5. Related Work

Recently, several works focused on how to build sound, complete, and robust web measurement studies. In 2020 Ahmad et al. observed that using a specific crawling technology significantly impacts the outcome of an experiment [4]. Furthermore, Demir et al. analyze the state of the art of how the web security and privacy community performs and documents their experiments [34]. They show that studies are often not reproducible or replicable—our work is a step towards solving this challenge. They provide guidelines that help design experiments free of such limitations. Jueckstock et al. show how different measurement tools and network access methods impact security and privacy measurements [87]. Their experiments show that the investigated parameters heavily impact, for example, "request/traffic volumes" or loaded JavaScript libraries. Cassel et al. analyze differences in observed tracking requests when using mobile or desktop browsers [25]. Yang and Yue develop *WTPatrol* and find in a comparative measurement study of web tracking on 23,310 websites with mobile version and desktop version webpages that mobile web tracking has unique characteristics. It became increasingly as prevalent as desktop web tracking in terms of the overall volume of trackers [179]. Vastel et al. find that 291 websites of the Alexa top 10k block crawlers effectively use fingerprinting [169]. Most recently, Calzavara et al. show that archive-based measurements might pose a reproducible solution to the reproducibility problem, and they develop best practices for future measurements [64]. Roth et al. find that seemingly static values like HTTP headers can differ based on the crawling location [135]. Finally, several other works discussed that small changes in a measurement setup could significantly affect the outcome (e.g., visiting subsites) [180, 107, 161, 4, 179, 8]. This work extends the existing body of research by providing an in-depth analysis of the effects of different measurement setups to understand their impact on websites' loading and content inclusion behavior.

3.3. Conclusion

In the following, we conclude our analysis of the results from Section 3.1 and Section 3.2 and discuss how our findings bolster the robustness of Web measurement studies.

Designing Web Measurement Studies. We have developed a set of best practices and 18 criteria for designing and conducting Web measurement studies to enhance their reproducibility and replicability, consequently increasing their robustness. In a large-scale measurement with multiple crawling profiles, we demonstrate that minor adjustments to the crawling technology (e.g., browser type or mimicking user interaction) may lead to significant differences in the results. We show that inadvertently documented experiments reduce the chances that researchers can reliably reproduce the results. More importantly, *replicability* and comparability of individual works cannot be universally assumed. A takeaway from this is that we as a community need to find ways to perform more robust Web measurements to draw reproducible and replicable conclusions from the conducted experiments.

Similarity of Web Measurements. In Section 3.2, we performed a large-scale web measurement study to understand the effects of different experimental setups commonly used to conduct such experiments. The results suggest that embedded first-party components show an almost perfect similarity, while third-party components and other consecutively loaded elements show much lower similarity values. Especially when we look at the loading dependencies, we see a substantial deviation between the profiles, indicating that the web's dynamic is an important factor one must consider when conducting and comparing web measurement studies. Furthermore, our results show that differences in the resulting dependencies exist—even if the same setup is used. From the perspective of privacy-related Web measurements, the differences in this context are critical because such studies often analyze phenomena primarily occurring in a third-party context.

In general, our findings highlight that we, as a community, must invest more efforts in researching and developing robust measurement setups to ensure the correctness of our experiments. Essential takeaways from our study in this direction are that:

(1) future work should investigate how to assess "measurement variances" in web experiments, which is standard in other disciplines;

(2) drawing conclusions based on loading dependencies is error-prone since they are often fluctuating;

(3) an understanding of if the phenomenon of interest is present in the dynamic (e.g., ads) or static (e.g., HTTP headers) content of a page is vital to plan the experiments; and

(4) our approach confirms that researchers should use different profiles and execute multiple measurements to assess the potential of ‘randomized’ findings.

Chapter 4

Security: Understanding Update Behavior of Websites and Its Impact on Security

In the previous Chapter, we discussed how to enhance the robustness of large-scale Web measurements and examined the reasons for the discrepancies in these measurements. We provided guidelines and insights for researchers to improve the robustness of their studies. Unlike other studies presented in this thesis, the measurement in this Chapter uses open-source data, demonstrating that utilizing such data sources is a perfect way to completely reproduce and replicate a Web measurement.

The measurement in this Chapter aims to understand the update behavior of software components on the Web and the potential implications of using outdated software. Based on the findings obtained in Chapter 3, this Chapter presents a Web security measurement as a case study. The results are based on 18 measurement points. Researchers, developers, and end users are given an overview of security-relevant trends on the Web and the significant impact of processing updates on Web security. It begins with an introduction in Section 4.1, followed by a presentation of the methodology used to identify software on webpages and measure their vulnerability in Section 4.2. Section 4.3 discusses the update behavior of software components in webpages and the impacts of utilizing outdated software on the websites' vulnerability. Based on our findings, we provide actionable recommendations in Section 4.4 for researchers, developers, and end users.[1]

4.1. Introduction

Nowadays, we use the Web for various tasks and services (e.g., talking to our friends, sharing ideas, to be entertained, or to work). Naturally, these services process a lot of personal and valuable data, which needs to be protected. Therefore, web services need to be hardened against adversaries, for example, due to imperfections of software. An

[1] The remainder of this Chapter is mainly based on the paper "Our (in)Secure Web: Understanding Update Behavior of Websites and Its Impact on Security", which was presented in the proceedings of the 22nd Passive and Active Network Measurement Conference 2021 (PAM) and co-authored by Tobias Urban, Kevin Wittek, and Norbert Pohlmann [39].

essential role in every application's security concept is the updating process of the used components [95]. Not updating software might have severe security implications. For example, the infamous *Equifax* data breach that affected 143 million people was possible because the company used software with a known vulnerability that has already been fixed in a newer version [181].

However, keeping software up to date is not always easy and, from the security perspective, not always necessary (i.e., not every update fixes a security issue). Modern applications require a variety of different technologies (e.g., libraries, web servers, databases, etc.) to operate. Updating one of these technologies might have unforeseeable effects and, therefore, updates might create potentially high overhead (e.g., if an update removes support of a used feature). More specifically, service providers might object to install an update because they do not directly profit from the new features (e.g., changes in an unused module). Hence, it is reasonable not always to install every available update (e.g., to ensure stability).

In this Chapter, we show that this challenge can have grave implications. To understand how up to date the utilized software on the Web is and to understand its possible security implications, we conduct a large-scale measurement. Previous work also analyzed update behavior on the Web (e.g., [155, 168]) but – to the best of our knowledge – our measurement is more comprehensive than the previous studies. While we analyze over 5.6M sites and nearly 250 software (SW) products, other work in this field often only analyzed one specific type of software or a small subset. Therefore, our results are more generalizable and provide a better overview of the scale of the problem.

To summarize, we make the following contributions:

- **Large-scale Web measurement.** We conduct a large-scale measurement that evaluates 246 software products used on 5.6M websites over a period of 18 months, to determine update behavior and security impact of not updating.
- **Understanding update behavior.** We show that 96% of the analyzed websites run outdated software, which is often more than four years old and is getting even older since no update is applied.
- **Understanding vulnerability of sites.** We show that a vast majority of the analyzed websites (95%) use software for which vulnerabilities have been reported, and the number of vulnerable websites is increasing over time.

4.2. Method

In this work, we want to assess the update behavior of web applications, measure if they use outdated software, and test the security implications of using the vulnerable software. To accomplish that, we collect the used modules (software and version) of the websites present in the *HTTPArchive* [67] over a period of 18 month, extract known vulnerabilities from the *National Vulnerability Database* database, and map them against the used software versions of the analyzed sites.

4.2.1. Identifying Used Software

To assess the update behavior of websites, we need to identify the software versions of the software in use. To do so, we utilized data provided by *HTTPArchive* [69], which includes all identified technologies used by a website. HTTPArchive crawls the landing page of millions of popular origins (mobile and desktop) based on the *Chrome User Experience Report* (CrUX) [60] every month, since January 2019. In CrUX, Google provides publicly metrics like load, interaction, layout stability of the websites that are visited by the Chrome web browser users on a monthly basis. This real-world dataset includes popular and unpopular websites [68]. In our study, we analyze all websites provided in HTTPArchive. Hence, we can use 18 data points in our measurement (M#1 – M#18). The data provided by HTTPArchive includes, among other data: (1) the date of the crawl, (2) the visited origins, and (3) identified technologies (software including its version). HTTPArchive uses *Wappalyzer* [171] to identify the used software, which uses different information provided by a site to infer the user version and technology stack. In order to make version changes comparable, we converted the provided data to the *semantic versioning* (SemVer) standard (i.e., `MAJOR.MINOR.PATCH`) [127] and validate also the version information from HTTPArchive as well as from NVD and check if provided versions are in a valid SemVer format. This unification allows us to map the observed versions of the known vulnerabilities. If we find an incomplete SemVer string, we extended it with ".0" until it fits the format.

4.2.2. Identifying Vulnerable Software

To better understand the security impact of updates, we map the software used by an origin to publicly known vulnerabilities. We collect the vulnerabilities from the *National Vulnerability Database* (NVD) [2]. Each entry in the NVD holds various information, but only three are essential to our study: (1) the date on which it was published, (2) a list of systems that are affected by it, and (3) the impact metrics how it can be exploited and its severity. In this work, we only focus on vectors that can be exploited by a remote network adversary.

4.2.3. Dataset Preparation and Enrichment

Here, we describe the steps taken to enrich our dataset to make it more reliable.

[2] We used the database published on 04/07/20.

4.2.3.1. Release History

To get a firm understanding of the update behaviour of websites, it is inevitable to know the dates on which different versions of a software were released ("release history"). To construct the list of release dates of each software product, we used *GitHub-API* for the official repositories, on *GitHub*, of the products and extracted the date on which a new version was pushed to it and store the corresponding SemVer. If a product did not provide an open repository on *GitHub*, we manually collected the official release dates from the product's official project webpages, if it's published.

4.2.3.2. Dataset Preparation

Since the Web is constantly evolving and Web measurements tend to be (strongly) impacted by noise, we only analyzed software products on a site for which we found version numbers in at least four consecutive measurements. Furthermore, we dropped all records with polluted data (e.g., blank, invalid versions, duplicates, dummy data) from our dataset. Finally, in order to make a valid match between CVE entries and software in our dataset, we manually assigned each software in our dataset their CPE (naming scheme) using the *CPE Dictionary* [115] provided by NVD.

4.2.4. Analyzing Updating Behavior & Security Implications

In this section, we describe how we measure update behavior and identify vulnerable websites.

4.2.4.1. Updating Behavior

To understand update behavior in our dataset, one needs to measure the deployed software's version changes over time. Utilizing the release dates of each software product, we know, at each measurement point in our dataset, whether a site/origin deploys the latest software version of a product or if it should be updated. If we found that an outdated product is used, we check if it was updated in the subsequent measurements (i.e., if the SemVer increases). This approach allows us to test if a product is updated after all and to check how long this process took. In our analysis, we call an increasing SemVer an *update* and decreasing version number a *downgrade*. In this analysis, we compare the `MINOR` and `PATCH` part of a product's SemVer, utilizing the release dates of each version, and not the `MAJOR` section because service providers might not use the latest major release due to significant migration overhead. For example, we would consider that an origin is "up to date" if it runs version 1.1.0 of a product even if version 2.1.0 (major release change) is available. However, if version 1.1.1 would be available, we consider it "out of date".

4.2.4.2. Identifying Vulnerable Websites

One way to measure the impact of an update on the security of a site is to test if more or less vulnerabilities exists for the new version, in contrast to the old version. To identify vulnerable software on a website, we retrieve the relevant CVEs for the identified software and then check if it is defined in these CVE entries – with consideration of *versionStart[Excluding/Including]* and *versionEnd[Excluding/Including]* settings. We map a vulnerability to a crawled origin if and only if (1) it uses a software for which a vulnerability exists and (2) if it was published before the crawl was conducted. Utilizing the *Common Vulnerability Scoring System* (CVSS) of each vulnerability, we can also assess the theoretical gain in security.

4.3. Results

After describing our approach to analyze the update behavior of websites and its possible security impact on websites, this section introduces the large-scale measurement results. Overall, we observed 8.315.260 origins on 5.655.939 distinct domains using 342 distinct software products. After filtering, we were left with 8.205.923 origins (99 %) on 5.604.657 domains (99 %) using 246 (72 %) software products. We collected 31.909 releases for 246 software products. Furthermore, we collected 147.312 vulnerabilities of which 2.793 (2 %) match to at least one identified product. Overall, we found an exploitable vulnerability for 148 (60 %) of the analyzed software products. Note that products with no public release history are excluded from analyzing update behavior and security analysis if they don't have a known vulnerability. Note also we have full access to all the segments of the `MAJOR.MINOR.PATCH` for 98.5% of our data. In total, we identified 12.062.618 software updates across all measurements. Table 4.1 provides an overview of all evaluated records of each measurement run.

4.3.1. Update Behavior on the Web

In the following, we analyze the impact of adoption of releases on the Web on website level and from software perspective.

4.3.1.1. Update Behavior of Websites

The first step to understand the update behavior of websites is to analyze the fraction of used software products that are fully patched, according to our definitions. Remember that we assume that a software product should be updated if a newer minor version or patch is available (i.e., we exclude the major version (see Section 4.2.4)). In our dataset, we identified a median of 3 (min: 1, max: 17, avg: 3.37) evaluable software products for each website. Overall, we identified that across all measurement points, on average, 94 %

Table 4.1.: Overview of all measurement points.

M.	Date	#Sites	#Origins	#Products	#dist. Ver.	#Updates	#Vuln.
M#1	01/19	2.5M	3.4M	208	15,436	—	2,201
M#2	02/19	2.3M	3.1M	204	15,178	0.4M	2,224
M#3	03/19	2.3M	3.1M	205	15,390	0.5M	2,235
M#4	04/19	2.7M	3.5M	205	16,145	0.6M	2,291
M#5	05/19	2.8M	3.6M	216	16,741	0.4M	2,298
M#6	06/19	2.8M	3.6M	217	17,013	0.7M	2,310
M#7	07/19	3.0M	3.9M	215	17,438	0.6M	2,286
M#8	08/19	3.0M	3.9M	215	17,474	0.5M	2,316
M#9	09/19	3.0M	3.9M	215	17,682	1.0M	2,390
M#10	10/19	3.0M	3.8M	217	17,873	0.8M	2,424
M#11	11/19	3.0M	3.8M	217	17,958	1.0M	2,468
M#12	12/19	3.0M	3.8M	216	18,122	1.0M	2,478
M#13	01/20	2.9M	3.8M	217	18,173	0.8M	2,502
M#14	02/20	2.7M	3.4M	211	17,558	0.4M	2,526
M#15	03/20	3.1M	3.9M	217	18,558	0.4M	2,412
M#16	04/20	3.3M	4.2M	217	19,321	0.6M	2,467
M#17	05/20	3.1M	4.0M	220	19,353	0.8M	2,460
M#18	06/20	3.4M	4.4M	218	20,118	0.6M	2,475

of all observed websites were *not* fully updated (i.e., at least for one software product exists a newer version). Only 6 % of the observed sites used only up to date software while 47 % entirely relied on outdated software types. The mean fraction of out of date software products is 74 % for each observed website across our measurement points. These numbers show that websites often utilize outdated software. While at domain granularity, almost all analyzed sites use outdated software, it is interesting to analyze if subdomains show different update behavior. Figure 4.1 compares the fraction of up to date software utilized on subdomains (e.g., *bar.foo.com*) against the root domains (e.g., *foo.com*), along our measurement points. In the figure, zero means that all software is up to date and one means that all software is outdated. Our data shows that most software products are not updated to the newest release, but it is still interesting to analyze the update cycles websites use in the field. On average, we observed 0.7M version changes between two measurement runs. 97 % of them were upgrades (i.e., the SemVer increased) and consequently 3 % were downgrades.

4.3.1.2. Update Behavior from a Software Perspective

Previously, we have shown that websites tend to use outdated software. In the following, we take a closer look at the used software to get a better understanding if the type of used software has an impact on its update frequency. Across all measurements, the software used on the live systems is 44 months old (M#1: 40, M#18: 48), and the trend during the measurement is that it gets even older (18 days each month on average). To determine

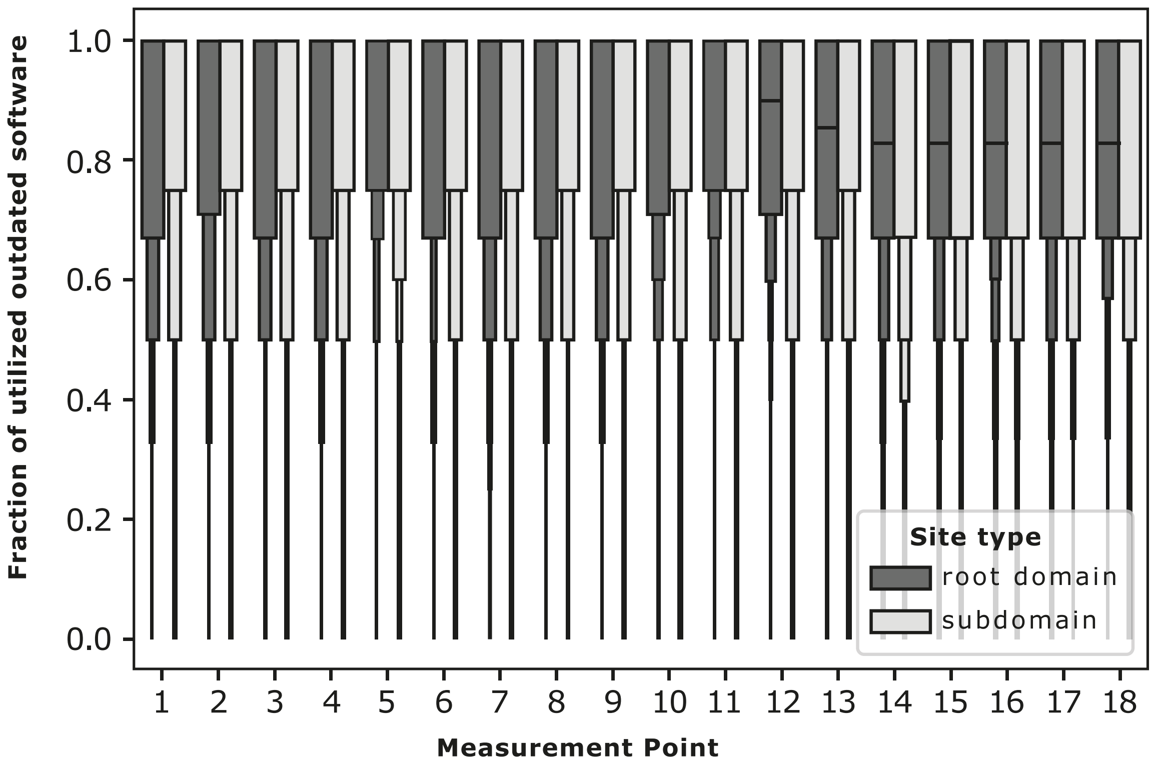

Figure 4.1.: Fractions of utilized outdated software products on the analyzed domains in comparison to their subdomains (1 = no product is up-to-date).

how the average age changes by software types, we measured the average age of the top ten used software types for all measurement points. These top ten account for 65 % of all analyzed software types. In Figure 4.2 we show the corresponding results. Our finding clarifies that client-side software (e.g., JavaScript Libraries) is older than server-side software (e.g., Web Servers). A closer observation of the releases SW shows that the server-side software has shorter release cycles than client-side software in the measured period (e.g., *nginx* has 18 and *jQuery* only has 6 releases). While the age of the software itself is not necessarily a problem per se, it is notable that the average number of months a utilized software is behind the latest patch is 48. The ANOVA test ($\alpha = 0.05$) showed no statistical evidence that the popularity of a website, according to the *Tranco* list [100], has an impact on the age of the used software (i.e., popular and less popular websites use outdated software alike). Using software that is four years old might be troubling, given that on average 41 newer version exists, because the software might have severe security issues. We have shown that overall mostly outdated software is used. However, it is interesting to understand if this applies to all types of software alike or if specific products are updated more frequently.

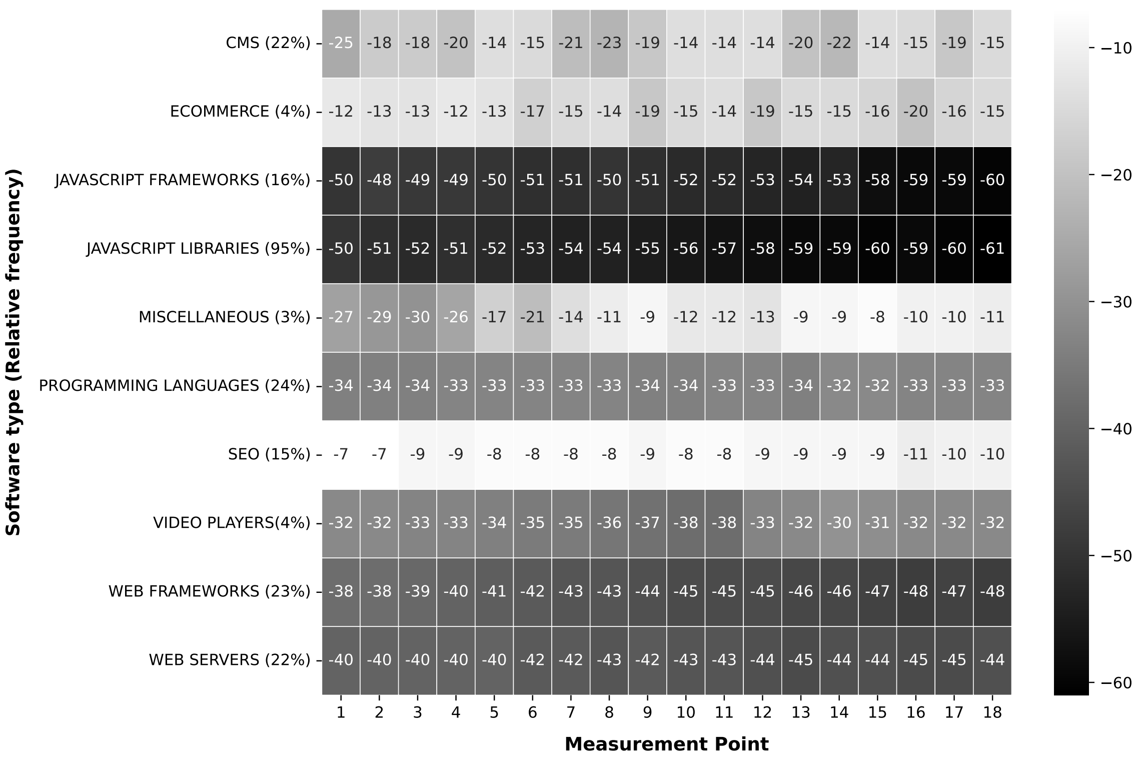

Figure 4.2.: Average age (in month) of top utilized 10 software types for all measurement points.

4.3.1.3. Adoption of Software Releases

To get a better understanding of the update behavior of websites, we observe the adoption of releases. We find that every month, on average, 67 % of the software used has a new release. However, our observations show that only a few service providers install the release promptly. We record that on average, only 7 % of available updates are processed (min: 4 %, max: 11 %). The mean time between two updates for any of the used software on one website is 3.5 months (SD: 5.4). To get a more in-depth understanding of the adoption of software releases, we measure it in a time span of 30 days after the release. Figure 4.3 shows the fraction of processed updates by websites in that time span for the top eight software types. The top eight types account for 60 % of all used software. In general, we see that PATCH level releases are processed most frequently. Furthermore, we observe that the adoption of release types differ based on the software types. E-Commerce software process PATCH releases most frequently and search engine optimization software (SEO) MAJOR releases respectively. We assume that integrated automatic background updates play an important role why specific software types are updated. For example *WordPress* and *Shopware*, two popular content management systems, provide an auto update functionality [177, 143].

Summary Based on our dataset, we have shown that the used software on the Web is often very old and not updated frequently. While differences in the update behavior

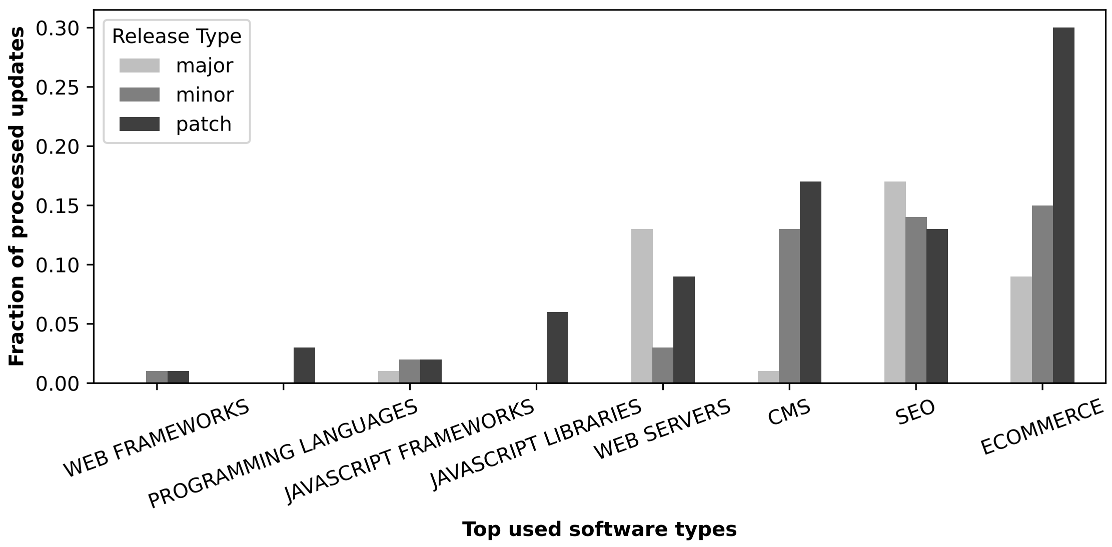

Figure 4.3.: Fraction of processing a new release for top used eight software types.

between different types of software exist, the majority of all times is still not updated. However, the impact of this not-updating is not clear and needs more investigation.

4.3.2. Security Impact of *Not* Updating

Experts agree that updating is one of the most critical tasks one should do to harden a system or to avoid data leaks [130]. Therefore, we are interested in the security impact of the identified tend to use outdated software.

4.3.2.1. Vulnerability of Websites

Towards understanding the threats that result from the usage of outdated software, we first analyze the scope of affected websites. On average, 94 % of the analyzed websites contain at least one potential vulnerable software, which was slightly increasing over the course of our measurements (M#1: 92 % to M#18: 95 %). We also record that each analyzed software has on average 8 vulnerabilities and that websites are affected, on average, by 29 (min: 0, max: 963). Our data shows that the number of exploitable vulnerabilities is decreasing over time for both per software (0.4 per month) and per websites (0.14 per month). Hence, overall the number of websites that have at least one vulnerability increases but the amount of vulnerabilities per site decreases.

Each vulnerability has a different security impact on a website, and, therefore, the number of identified vulnerabilities does not directly imply the severeness of them. The NVD assigns a score to each vulnerability to highlight its severeness (i.e., the CVSS score). Figure 4.4 shows the mean CVSS scores for the analyzes websites their rank. By inspecting the figure, one can see that less popular sites (the rank is higher) are affected by more severe

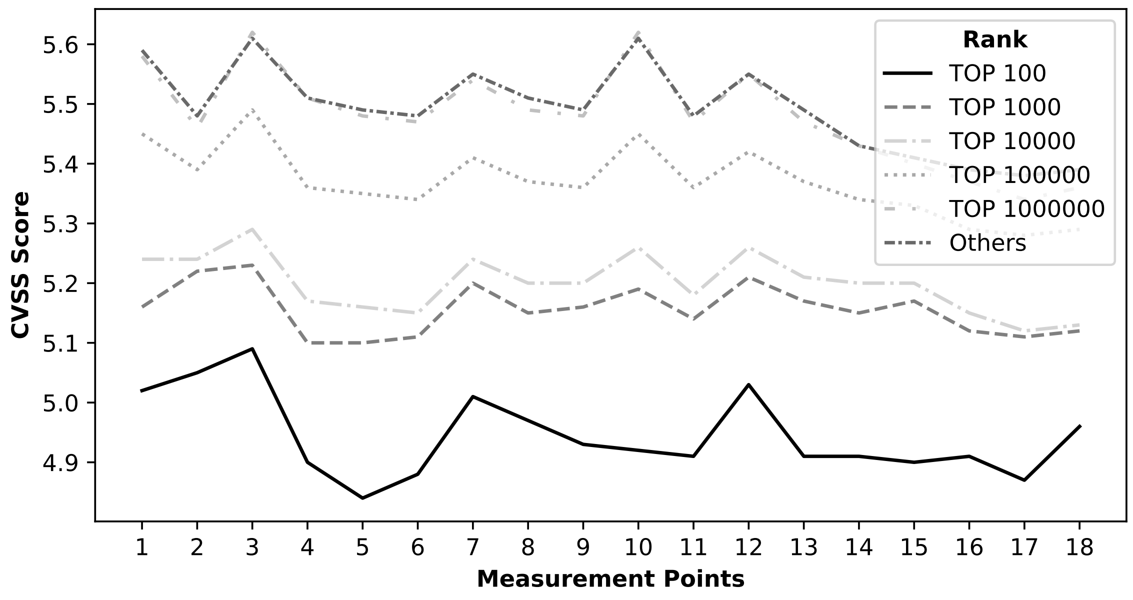

Figure 4.4.: AVG CVSS by popularity of websites. Vulnerability severity is significantly lower for high-ranked websites.

vulnerabilities. The Spearman test ($\alpha = 0.05$) showed a statistical significant correlation between the rank and the mean CVSS score of the identified vulnerabilities (p-value < 0.007). Table A.2, in Appendix A.3.1 lists the most common vulnerabilities in our last measurement point (M#18). A stunning majority of websites (92 %) is theoretically vulnerable to *Cross-site Scripting* (XSS) attacks. In our dataset, *jQuery* is the software that is most often affected by a CVE (92 %). A list of the most prominent CVEs is given in Appendix A.3.3. Given the wide occurrence of vulnerabilities in our dataset, the question arises which threats websites and users actually face.

4.3.2.2. Analysis of Available Vulnerabilities

Figure 4.5, shows the distribution of severity of identified vulnerabilities on websites based on the *Common Vulnerability Scoring System* (CVSS). Our results show that the number of websites with the most severe vulnerability (CVSS: 10) steadily decreases. The average number of vulnerable websites with a severity “HIGH” (CVSS: 7–10) is decreasing (M#1: 43 %, M#18: 39 %), while the number of vulnerable websites with “MEDIUM” (CVSS: 4–7) remains almost constant (M#1: 47 %, M#18: 49 %). For this analysis, we only used the most severe vulnerability for each website.

Given the result that the average age of used software depends on its type (see 4.2), we find that older software has more dangerous vulnerabilities. For example, the average CVSS/age of JavaScript-Frameworks was 4/50 in M#1 and 6/62 in M#18, while the score and age for programming languages go from 9/34 to 8/33. This confirms that older software *does* have more vulnerabilities and highlights the need for better update processes of websites. Furthermore, our analysis shows that performing updates has a significant impact on

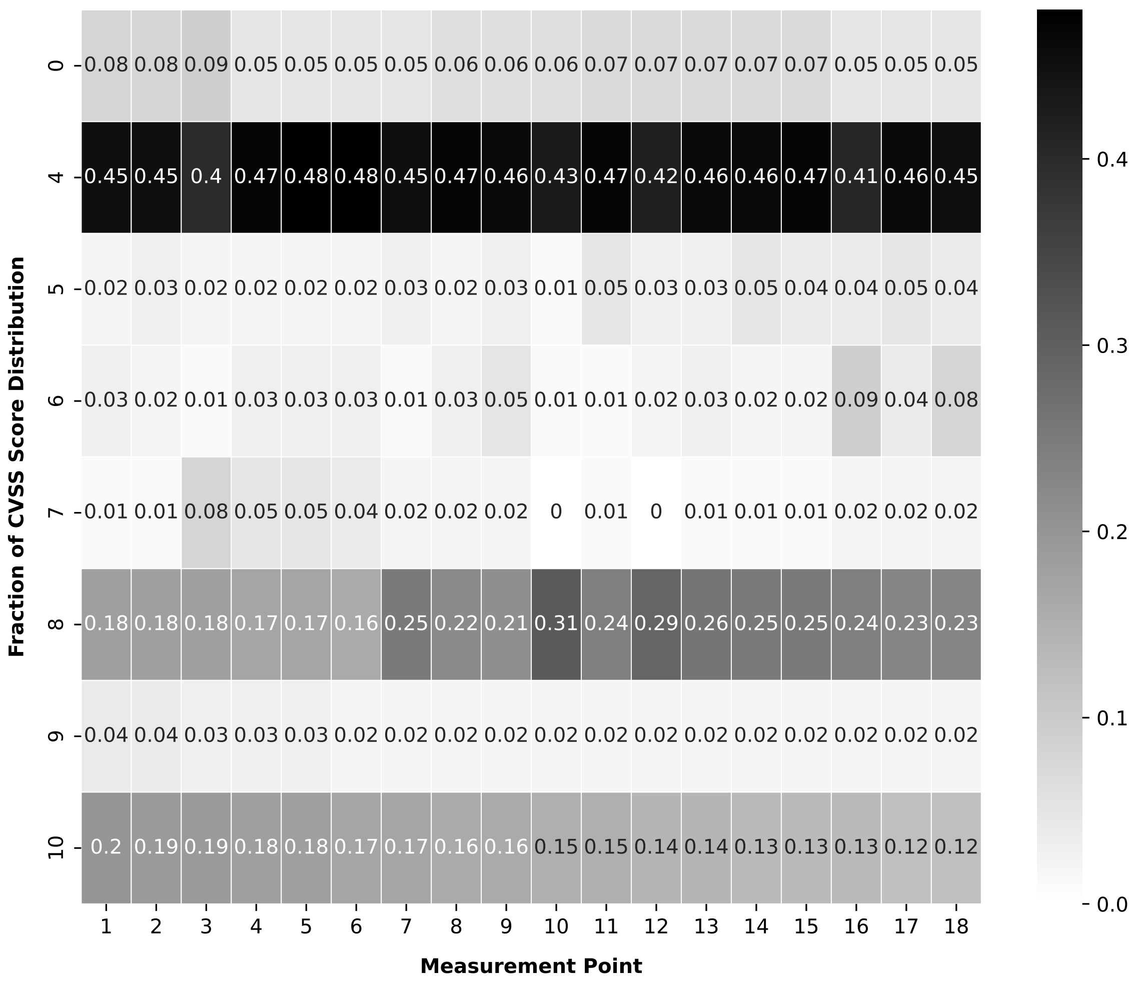

Figure 4.5.: Fraction of CVSS score distribution on websites for all measurement points (10 = Critical, 0=No Vulnerability).

the security of software. The average value of CVSS for software for which an update is available is 6.4 ("MEDIUM"). However, after applying the update(s), the CVSS is lowered to 2.4 ("LOW").

4.4. Recommendations

In this Section, we provide actionable recommendations for researchers, developers, and end users based on our results.

4.4.1. Recommendations for Researchers

Researchers should consider using open-source data in their studies to enhance reproducibility and replicability, provided it aligns with their research objectives. Open-source data is publicly accessible, which makes it easier for other researchers to validate the findings and conduct follow-up studies. However, researchers should be aware of the limitations of open-source data by studying methodologies of open-source data.

Recommendation #1:

Researchers should leverage open-source data for increased study reproducibility and replicability, while being aware of its limitations and potential impacts.

We have observed that websites often employ varying versions of software, many of which are outdated, significantly impacting the website's vulnerability and overall security. In light of these findings, researchers should be mindful of the potential effects of differing software versions on their results. It is important to investigate further whether these variations influence the measured metrics, as this could affect the accuracy of the research outcomes.

Recommendation #2:

When conducting web measurements, researchers should consider how different software versions on webpages impact the measured metrics.

4.4.2. Recommendations for Developers

Developers should enable auto-updates for the software components in their Web environment. By automating the update process, developers can ensure that their applications are always running the latest versions of software components with the most recent security patches and bug fixes. This reduces the risk of exploitation by malicious actors and enhances the overall security and performance of the Web environment. Additionally, it minimizes the manual effort required to maintain and update the software.

Recommendation #3:

Developers should enable auto-updates to ensure their web applications run the latest software versions, and minimizing manual maintenance effort.

Developers should adopt *Semantic Versioning (SemVer)* for their software. *SemVer*.By adopting *SemVer*, developers can more easily match software versions across different contexts by comparing third-party software versions with locally used ones or matching a version in a *CVE* entry with the utilized software. In addition, it facilitates subscribing to security advisories and vulnerability databases, which is vital for staying informed about potential security risks. This can enhance the compatibility and security management of software components.

Recommendation #4:

Developers should adopt *Semantic Versioning (SemVer)* to easily match software versions across contexts and enhance compatibility and security management.

Developers should implement auto-update mechanisms for their web applications to streamline the update process. By automating updates, developers can ensure that their users [3] always have access to security enhancements without requiring manual intervention. Auto-update mechanisms can be implemented using various technologies and approaches, such as web service workers or application update APIs. Implementing auto-updates simplifies the user experience, reduces friction, and enhances the application's overall security.

Recommendation #5:

Developers should implement auto-update mechanisms in software to ensure users always receive security updates, simplifying the user experience and boosting overall security.

When releasing updates, developers should consider backward compatibility to prevent disruptions. Ensuring new software versions are compatible with previous versions helps transition smoothly without encountering issues. Developers should avoid making breaking changes that would require to make significant adjustments or modifications to their existing workflows. By prioritizing backward compatibility, developers can provide an enhanced experience and maintain trust during the update process, enhancing overall security.

Recommendation #6:

Developers should prioritize backward compatibility in software updates to ensure smooth transitions, prevent disruptions, and enhancing overall security.

4.4.3. Recommendations for Users

Users [4] should be aware that almost all websites on the Web have vulnerable software components (cf. Section 4.3.2), which may expose them to various risks such as data

[3] In this section, 'Users' refers to users of software components, and not visitors of webpages.
[4] In this section, 'Users' refers to visitors of webpages.

breaches, malware, or phishing attacks. Users should exercise caution when providing personal or sensitive information on websites. By staying informed about the potential vulnerabilities of websites, users can better protect themselves and their data from potential threats. Additionally, users need to educate themselves on recognizing signs of secure websites, such as HTTPS in the URL, valid security certificates, checking the modernity of websites, and trusted website seals that inform users about the website's vulnerability.

Recommendation #7: ✓

Users should be aware that most websites have vulnerable software and exercise caution when sharing personal information, checking for signs of website security.

Using modern web browsers is crucial for ensuring a safe browsing experience. Modern browsers are equipped with built-in security features that protect users from malicious websites, phishing attempts, and other threats. They also receive regular updates that address security vulnerabilities, improve performance, and add new features to enhance the user experience.

Recommendation #8: ✓

Users can benefit from enhanced security measures that protect their personal information and online activities by choosing to use up-to-date web browsers.

Popular websites often have more resources to invest in their infrastructure, resulting in better performance, security, and user experience. Moreover, well-known websites are more likely to be scrutinized by the security community and the public, encouraging them to prioritize the safety of their users. In contrast, less-popular sites may have a different level of resources to maintain a secure and optimized web environment, making them more susceptible to threats which our results also show. Users can generally expect a more reliable and secure browsing experience by opting for popular sites.

Recommendation #9: ✓

Users should opt for popular websites, as they often offer better security and performance due to higher scrutiny and resource availability.

4.5. Limitations and Ethics

Although we have put a great effort while preparing our dataset, our study is impacted by certain limitations. Our approach comes with the limitation that, on the one hand, *HTTPArchive* only crawls landing pages and does not interact with the website, which might hide the complexity of an origin [161], and, on the other hand, *Wappalyzer* might not detect all used software for the website. Although NVD is one of the most popular vulnerability databases, there are some discussions around the accuracy of the data provided by NVD e.g., [116, 142]. In our study, we assume that software utilized by a website is vulnerable if the NVD provides a CVE entry for it. For ethical reasons, we did not validate if successful exploitation of the CVE requires any interaction or enabled functions. We also do not examine any mechanism for the validity of CVE entries.

4.6. Related Work

To the best of our knowledge, our study is the first one that measures update behavior and security implications by evaluating all utilized server and client-side software on a website and by conducting multiple measurements. In the following, we discuss studies related to our research.

Update Behavior The software update behavior has been previously studied. Tajalizadehkhoob et al. [155] measure the security state of software provided by hosting providers to understand the role of hosting providers for securing websites. Vaniea et al. [168] conduct a survey to understand the update behaviour of software. They ask 307 survey respondents to provide software update stories and analyze these stories to determine the possible motivations for software updates. Stock et al. [152] examine the top 500 websites per year between 1997 and 2016 utilizing archive.org dataset. In their measurement, they mainly evaluate security headers and analyse usage of outdated *jQuery* libraries.

Security Implications Prior literature has proposed various techniques to measure websites' security in terms of different metrics. Lauinger et al. [99] study widely used 72 client-side JavaScript libraries usage and measure security across Alexa Top 75k. Van Goethem et al. [57] report the state of security for 22,000 websites that originate in 28 European countries. Their analysis is based on different metrics (e.g., security headers, information leakage, outdated software). However, they use only a few popular software products for their measurement. Huang et al. [73] measure the security mechanisms of 57,112 chinese websites based on vulnerabilities published on Chinese bug bounty platforms between 2012 and 2015. Van Acker et al. [165] scrutinize the security state of login webpages by attacking login pages of websites in the Alexa top 100k.

4.7. Discussion and Conclusion

In this work, we measured the update behavior and possible security implications of software products utilized on more than 5.6M websites. Our measurement highlights the current state of the Web and shows the update behavior of websites over the course of

18 month. We show that most of the Web's utilized software is outdated, often by more than four years. Running outdated software is not a security problem per se because the old software might not be vulnerable. However, we found several sites that use software products for which vulnerabilities have been reported. Furthermore, we show that the number of vulnerable websites increases over time while the average severity of identified vulnerabilities decreases. For instance, we record that 95 % of websites *potentially* contain at least one vulnerable software. It has to be noted that the identified vulnerabilities in our work must be seen as an upper bound because utilizing a product for which vulnerabilities exist does not automatically mean that it can be exploited (e.g., the vulnerable module of the product is deactivated or not used). Our results still highlight that website providers need to take more care about their update processes, even if this comes with a potential overhead, to protect their users and services.

Chapter 5

Privacy: A Large-Scale Study of Cookie Banner Interaction Tools and Their Impact on Users' Privacy

In the previous Chapter, we performed a case study on Web security, based on the findings from Chapter 3 and explored the update practices of providers and their impact on website security. In this Chapter, we conduct another large-scale Web measurement. Once again, we apply the findings from Chapter 3 to enhance the robustness of this study.

We investigate the efficiency of various Web privacy tools and potential side effects that could adversely affect end user privacy instead of enhancing it. We offer actionable recommendations for end users, developers, and researchers based on our results. We begin with an introduction in Section 5.1 and present our methodology for this study in Section 5.2. We then present our findings in Section 5.3. Subsequently, we offer recommendations in Section 5.4 for researchers, developers, and end users.[1]

5.1. Introduction

Websites make rich use of HTTP cookies for various means (e.g., user tracking). To provide (European) Internet users more control over the usage of cookies, many website providers embed so-called "cookie banners" on their webpages [52, 33, 66]. Some banners allow fine-grained control over the type of cookies to be used (e.g., rejecting advertising cookies), while others only inform users about their usage [163]. Cookie banners are meant to help users but are often designed in a way that nudges the user to accept all types of cookies rather than to reject them ("Dark Patterns" [63, 93]), and their the omnipresence has started to annoy users [96].

Different tools help users cope with cookie banners by automatizing the interaction process [18, 119, 118, 154, 89]. These tools are usually implemented as a browser extension and often use rule-based approaches to identify and interact with banners. More specifically,

[1] The remainder of this Chapter is mainly based on the paper "A Large-Scale Study of Cookie Banner Interaction Tools and Their Impact on Users' Privacy", which stands to be presented in the proceedings of the 23rd Privacy Enhancing Technologies Symposium 2024 (PETS) and is co-authored by Tobias Urban, Norbert Pohlmann, and Christian Wressnegger [38].

the tools identify the banners and corresponding buttons on predefined patterns, similar to ad blockers that block URLs based on filter lists. Some tools offer the option to choose the cookie type the user consents to be set (e.g., "functional cookies" only). Users hence need to configure these tools according to their own privacy needs. However, the impact of these tools on the user's privacy still needs to be better understood. One challenge is that the interaction with these banners is neither standardized nor is it (legally) defined what the purpose of a cookie is or how it can be determined. Thus, while users rely on these tools for convenience and to implement their choice, it is still being determined if these tools even meet these expectations and which impact the tools have on users' privacy.

In this work, we perform a measurement study to understand the effects of six extensions for "cookie banner interaction." We analyze five tools used in the field and one custom extension developed for this study. In our experiments, we investigate (1) which and how many cookies a website sets, (2) the purpose of the used cookies, and (3) various deferred effects, such as the impact on tracking requests. With this study, we aim to understand the impact and potential benefits of the different tools for users' privacy. To do so, we visit 298k distinct pages on 30k websites once with each tool and uncover statistically significant differences in the analyzed extensions' effectiveness. We show that the number of cookies, category of used cookies, and individual cookies in terms of their key names differ based on the used extension.

Previous studies have either analyzed the design of different banners [163, 33, 159, 63] or have built new tools to interact with banners in a meaningful way [18, 119]. While we build upon prior work by utilizing some of the presented tools in this work, we investigate an entirely different problem: The effectiveness of different tools that automatically interact with cookie banners. To the best of our knowledge, this is the first work comparing the impact of "cookie banner interaction" tools on the user's privacy at a large scale.

In summary, the main contributions of this work are:

- **Large-scale measurement study.** We compare the effectiveness of six different tools that assist users in the interaction with cookie banners and show that the tools can impact the usage of the setting of specific types of cookies by up to 640%.
- **Tool effectiveness varies largely.** The results show that although some tools claim to have similar capabilities (e.g., accepting functional cookies), several differences exist in the number and type of observed cookies.
- **Effects of interacting with cookie banners.** We find that statistically significant effects of the interaction with cookie banners exist. For example, the number of tracking requests increases by up to 60% if an extension allows the usage of all cookies.

5.2. Method

The measurement framework provided by [34] is the basis for our setup, which allows orchestrating quasi-parallel web measurements using different browser setups. The framework consists of a master machine that starts multiple virtual machines (VM), each running a separate crawler with a distinct configuration. We use the individual configurations to implement nine different crawling profiles (cf. Section 5.2.4) for our experiments.

The *Google Cloud Platform* hosts our VMs, and all but one use an IP address associated with a European server (Frankfurt am Main; DEU). Each profile is based on *OpenWPM* [44] (`v0.20.0`), uses the *Firefox* browser (`v100.0` with the default user agent[2] and a screen resolution of `1920x1080`), and each profile uses a different browser extension that interacts with a cookie banner (cf. Section 5.2.2). We configure *OpenWPM* to log the entire HTTP(s) traffic, all cookie interactions (e.g., creation or deletion), and data stored in the local storage of the browser [109]. Since *OpenWPM* natively does not support the extraction of this storage, we implemented a custom function that extracts the values from this storage. We make our crawling setup publicly available (cf. Appendix A.1).

A crawl of the Web can either be *stateless* or *stateful*. Stateful crawlers keep the state of the browser between page visits (like the browser of a real user would), while in stateless crawls, the browser is reset between two page visits. In our experiment, we perform a stateful crawl for each visited site, meaning we keep the browser's state when visiting a site's pages. The state is reset when a new site is analyzed. This design choice has major implications for our experiment because after interaction with the cookie banner on the first page (typically the landing page of a site), the banner might not show on other pages and, therefore, we can record the behavior of a site after the user made a choice. However, the order of visiting sites might impact the resulting cookie jar.

To allow fair comparability of the different profiles, we only include sites in our analysis if at least eight profiles successfully crawled all site pages in our dataset. More specifically, if at least two profiles did not crawl a given page, the entire site is dropped from our analysis. This filtering applies to 10 888 (36%) of the sites in our dataset that we excluded from our analysis. Since we visit the sites in parallel, this approach allows us to keep the data we compare consistent. More precisely, we only compare sites if the underlying data was collected from the same pages. We elaborate on this seemingly high exclusion rate in Appendix A.4.2.

5.2.1. Dataset

To build the set of pages to visit, we resort to the widely used quasi-standard Tranco list [100] generated on 29/08/2022 [3]. From the list, we randomly sampled a subset of sites to analyze based on their ranking. We divided the list into buckets and randomly drew sites from each bucket. We used the top 5,000 sites from the list and randomly selected 5,000 sites from each of the following rank 'buckets': 5,001–10k, 10,001–50k, 50,001–250k, 250,001–500k, and 500,001–1M. Thus, in total, we used 30,000 sites for our experiment. We

[2] `Mozilla/5.0 (X11; Linux x86_64; rv:100.0) Gecko/20100101 Firefox/100.0`

[3] `https://tranco-list.eu/download/X568N/1000000`

Table 5.1.: Overview of the used extensions that claim to interact with cookie banners automatically. The date indicates the last update of the extension. User numbers as of 02/23.

#	Name	Date	Firefox Users	Chrome Users	Num. Configs
I	I don't care about cookies	02/23	317 000 +	900 000 +	1
II	Consent-O-Matic	02/23	21 000 +	60 000 +	7
III	Ninja Cookie	04/21	6000 +	40 000 +	1
IV	SuperAgent	06/22	3000 +	20 000 +	5
V	CookieBlock	08/22	2000 +	5000 +	4

use this sampling to understand if there are differences in the efficiency of the analyzed tools based on a site's popularity (rank).

Previous work has shown that subpages behave differently than the respective landing page [161, 34, 8]. Therefore, to build the final set of pages to visit, we visit each randomly selected site's landing page and collect 15 subpages (i.e., first-party links on the page) for each. We repeated the process recursively if the landing page did not hold enough links. We chose to use 15 pages since previous work showed that this number is a fair trade-off between crawling time and still capturing the behavior of a site. Overall, we visit 298k pages on the 30 000 sites utilizing the nine measurement profiles (on average 288k per profile). We provide a list of all analyzed pages and sites in the supplementary material of this paper (cf. Appendix A.1).

5.2.2. Cookie Banner Interaction Tools

To understand the impact of different tools that assist users in interacting with cookie banners, we perform a measurement study and compare the effects of other tools. Table 5.1 lists the extensions we analyze in this study. All analyzed extensions are supported by the most common browsers (e.g., Firefox, Chrome, or Safari). We chose these extensions since they are either popular (e.g., *I don't care about cookies* and *Ninja Cookie*) or were recently proposed by academic works as solutions to interact with cookie banners (e.g., *Consent-O-Matic* [120] and *CookieBlock* [18]). In our measurement, we install each extension in a separate *OpenWPM* instance, allowing us to compare each tool's effects individually. It should be noted that there is no specification (or understanding) of how cookie banners should work and not even a (legal) definition of how to define the type or purpose of a cookie. Yet, the extensions claim that they provide privacy or let users take control of their privacy and eliminate the need to interact with cookie banners. Therefore, it is interesting to understand their impact on users' privacy.

I don't care about cookies. The extension is by far the most popular browser extension in our measurement. The authors of the extension claim that it "*removes cookie warnings from almost all websites*" [89]. The tool's main intention is to be not to protect users' privacy but

Table 5.2.: Overview of the measurement profiles. Profiles #3 and #7 are custom extensions as described in Section 5.2.4. The last column marks tools that can process all banner types (and do not state any limitations of banner interaction) with a ✓.

#	Ext.	Ver.	Loc.	Method	Cookie Policy	All Banners
1	I don't care about cookies	3.4.2	DE	rule-based	Hides banners, accepts/rejects	✓
2	Consent-O-Matic	1.0.8	DE	rule-based	Reject all	Only CMPs
3	Consent-O-Matic *(custom)*	1.0.8	DE	rule-based	Allow all	Only CMPs
4	Ninja Cookie	0.2.7	DE	rule-based	Reject all	✓
5	CookieBlock	1.1.0	DE	ML-based	Accepts functional	✓
6	SuperAgent	2.6.0	DE	rule-based	Accepts functional, perf.	✓
7	"Accept all" Ext. *(custom)*	—	DE	rule-based	Accepts all	✓
8	None	—	DE	—	No interaction	—
9	None	—	US	—	No interaction	—

to eliminate the hassle of interacting with the banner. For example, the extension hides the HTML element(s) that compose the cookie banner. If that is not possible, the extension will interact with the banner by accepting all, some, or no cookies. The extension's description does not elaborate on how it determines its action. However, manual inspection and consultation with the developer showed that the extension usually accepts all cookies.

Consent-O-Matic. This extension was developed as part of a research project [120]. *Consent-O-Matic* was developed to interact with banners provided by different Consent Management Providers (CMPs) and aims to choose the most privacy-friendly setting by default. The extension works for a fixed set of CMPs and uses rules to interact with each banner. While, by default, the extension only accepts "necessary" cookies, it can also be configured to accept other types of cookies. We use this option to compare two extension configurations (cf. Section 5.2.4). When conducting our experiments, the extension supports 36 CMPs, which means if a website uses a banner not provided by any of those CMPs, the extension will not do anything. Previous work has shown that only roughly 11% of all websites use a cookie banner provided by a CMP [172, 18, 20].

Ninja Cookie. By default, this extension will interact with cookie banners and only allow the setting of essential cookies [118]. No public documentation exists on how *Ninja Cookie* identifies the cookie banners. However, by manually inspecting the source code, we found that it works on a heuristic, ruled-based approach to identify and interact with a banner. According to the documentation, the tool also works for major CMPs and other banners. *Ninja Cookie* is the only extension that offers a premium subscription that provides additional features such as whitelisting. In our experiment, we use the basic subscription, which is free.

CookieBlock. This extension was also developed as part of a research project [18]. Since the work was published in 2022, the extension only comes with low installation numbers. We still integrated the extension into our experiment because it is the only tool not relying on a rule-based approach but using machine learning. Furthermore, the extension does not aim to interact with cookie banners but automatically deletes cookies that the user does not desire. By default, the extension allows setting "functional" cookies. In our experiments, we use this default configuration of *CookieBlock*.

Super Agent. The last extension in our corpus is *Super Agent* [154]. Like most solutions, this extension uses a rule-based approach to identify and interact with cookie banners. The extension's default settings allow the setting of "functional" and "performance" cookies. In our experiment, we use these default settings.

5.2.3. Manual Extension Verification

We begin by testing the functionality of the extensions to understand whether they interact with cookie banners and thus work as intended. To this end, we randomly sample 19 sites from each of the used buckets that show a cookie banner and one that does *not* show a banner. In the latter, we include testing if the extensions might break a page's function. First, we manually visited each of the selected sites and tested if they showed or did not show a cookie banner using a vanilla Firefox browser and a European IP address associated with a German university (`redacted`; DEU) to gather ground truth. Then, we use the same setup (vanilla Firefox browser and a German IP address) and (1) manually install each extension separately, (2) visit the landing page of each site, and (3) check if the extension interacts with the banner.

Overall, we find that each of the identified extensions interacts, on average, with 12 (65%) (SD: 21%; max: 95% min: 48%) of all banners. Appendix A.4.1 presents a more detailed overview of the manual verification of each extension. Since all extensions can interact with some cookie banners, we include all of them in our subsequent experiments.

5.2.4. Profiles Used in the Measurement

In our experiment, we use nine profiles to analyze the effects of tools that assist users in interacting with cookie banners. For each of the previously introduced tools (cf. Section 5.2.2), we create an individual measurement profile, that is, a distinct *OpenWPM* instance that installs one of the extensions to the Firefox browser. In all but one profile, we use the default configuration for all extensions. In the only exception, we configure *Consent-O-Matic* to accept all cookies, which rejects all non-necessary cookies by default.

In addition to the named browser extensions, we used three additional browser profiles in our measurement. The first two profiles are plain *OpenWPM* installations that we use as reference values. These baselines allow us to compare how each website behaves if no extension is used, allowing us to understand the magnitude of the effect of the analyzed extensions. It is important to note that these baselines are not supposed to serve as a lower or upper bound but only as a reference to understand how a website utilizes cookies if we do not interact with a cookie banner. Like all other profiles, one is run from a German IP address (Frankfurt am Main; DEU), and the other is run from a US IP address (Council Bluffs, IA; USA). For the third profile, we use a custom rule set and a heuristic approach to 'click' the "accept all cookies" button. We provide details on this implementation in the following paragraph. This profile is supposed to provide an upper bound on the cookies set if users accept all of them. Table 5.2 shows an overview of all used profiles.

Custom "Accept All" Extension. To understand an upper bound of used cookies, we implemented a tool that aims to automatically "accept all" cookies of a page by triggering the respective button on a cookie banner. In the following, we briefly describe our approach. We implemented a custom command for *OpenWPM* that uses a two-step process to identify "accept all" buttons, following a heuristic signature-based approach: (1) Identify an accept-all button and (2) check if the button is used in a privacy-related context.

First, we analyzed the top 1,000 sites from the Tranco list to generate a signature list to identify accept-all buttons. We manually identified 483 sites that use a consent banner and collected 97 distinct accept-all button labels. While these numbers seem low at first sight, they are in line (and even exceed) findings on the average number of sites that use cookie banners [172]. Second, to determine if a button we identified in step one is used to manage consent, we profit from privacy-related words presented by [33]. After a page finishes loading, our extension tries to find the 'accept all' button in the DOM object using the created signature list. Afterward, we test if the parent elements of the identified buttons contain privacy-related keywords (Step 2). While this approach has several limitations, we still think it is viable to serve as an upper bound for our experiment. In our experiment, we could interact with cookie banners on 17% (5173) of the sites using this approach[4]. Our extension is publicly available (cf. Appendix A.1).

5.2.5. Identifying the Purposes of Cookies

This work investigates the effectiveness of browser extensions that interact with cookie banners. To better understand what kind of cookies are set in the different profiles, we classify them according to their purpose. In this work, we use four types of cookie categories proposed by the *International Chamber of Commerce UK* [77], which have been used by prior work [161, 18]. These categories are: (1) "Strictly Necessary Cookies" needed to provide the basic functionality of a website, (2) "Performance Cookies" aggregate (anonymously) the user's usage of the website, (3) "Functionality Cookies" personalize the website's usage, and (4) "Targeting/Advertising Cookies" used to track users or to display personalized ads.

[4] These number relates to *all* sites and not only those that use a cookie banner.

To identify the categories, we profit from *Cookiepedia* [28], a pre-categorized cookies database. *Cookiepedia* is run by *OneTrust*, a privacy management software company that also provides a CMP. The database provides for a given cookie name (i.e., the key) with additional data on the cookie. For example, for the cookie "`__cf_gads`," the database provides that it is used for "Targeting/Advertising" by *DoubleClick*. In our experiment, we query the database for additional data on each identified cookie. This process might be error-prone as cookie classes are assigned by hand but are—from our point of view—the best approximation of cookie usage today. We have been able to classify 57% of all observed cookies. Appendix A.4.7 provides an overview of the cookies that could not be classified.

In our experiment, we only focus on cookies in the cookie jar after crawling the last page for a given site; note our stateful crawling approach. We do this because some extensions (e.g., *CookieBlock*) might delete cookies from a category that the users opted out to. Hence, some cookies might be set by a page and deleted by an extension. In these cases, we assume the extension works as intended since undesired cookies are not persistent on the client. We evaluate the final set of cookies present by comprehending the actions logged by *OpenWPM* (i.e., creation, updating, and deletion).

We use *EasyList* to understand the effects of the analyzed extensions on user tracking. The list is a crowd-sourced effort to identify web tracking and might be incomplete or inaccurate to some extent. However, we expect that this limitation has only a marginal impact on our results and does not endanger our experiment's correctness.

5.2.6. Statistical Comparison of Profiles

We use the *Jaccard index* to compare cookies on a page or site. The index is used to gauge the similarity of sets and is defined as follows: $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$. By design, the index ranges from 0 to 1, where 1 denotes that the sets are equals and 0 that they have no element in common. Using the index, we can compare the similarity of the cookies set by a page and the impact of the used extension.

In our analysis, we use statistical tests to determine if an extension significantly impacts the usage of cookies. In particular, we use the *non-parametric permutational multivariate analysis of variance* (PERMANOVA) [7] with $\alpha = .05$ to find statistically significant differences between the measures of independent groups. In our case, these groups are, for instance, the used profiles. Furthermore, we utilize the permutation test [175] (10,000 random permutations) to understand if there is or is not a difference between treatment groups (i.e., the extensions). Finally, we use the *Kruskal-Wallis* test ($\alpha = .05$) to assess if samples originate from the same distribution. Due to their non-parametric nature and flexibility, these three tests allow us to robustly verify our hypothesis without relying on specific distributional assumptions.

5.3. Results

This section provides an overview of the measurement results. First, we provide a general overview of our measurement (Section 5.3.1), then we analyze the effect of the used extensions in terms of the number of cookies (Section 5.3.2), type of used cookies (Section 5.3.3), changes in the usage patterns of cookies (Section 5.3.4), and impact of the rank of a site (Section 5.3.5). Finally, we investigate the impact of the extensions on objects in the *HTML Web Storage* (Section 5.3.6) and analyze potential subsequent effects of the extensions (Section 5.3.7).

5.3.1. General Measurement Overview

In our measurement, we successfully crawled 29,660 (99%) sites, and 2.6M pages. On average, each crawler successfully visited 288k (SD: 5,180; min: 281k; max: 297k) webpages. We successfully crawled 12 (SD: 7; min: 1; max: 16) pages per site, on average (the landing page plus at most 15 subpages). The sites our crawlers could not reach are not meant to be visited by a human (e.g., link shorteners or content delivery networks). As already described, we include only sites in the evaluation successfully crawled by at least eight profiles (cf. Section 5.2). In our measurement, that filtering corresponds to 18,445 (62%) of the sites. The timing difference between two page visits across all profiles is 8 seconds (SD: 231; min: 1; max: 843), on average. The resulting (raw) data has a size of roughly 1.5 TB, which we make openly available (cf. Appendix A.1).

Overall, we record 415k (SD: 153k; min: 181k; max: 770k) cookies per profile on average. Using *Cookiepedia*, we could classify 57% of all cookies and 38% of all distinct cookies in our dataset (cf. Section 5.2.5). Figure 5.1 presents a general overview of the number of cookies set per page of each profile in our measurement setup. By visual inspection, one can see that some of the analyzed extensions substantially impact the setting of cookies in each profile. Thus, each extension impacts users' privacy differently. The permutation test found a statistically significant (p-value < 0.001) impact of the profile on the number of cookies. However, the same impact was *not* observed when comparing profiles #2, #4, and #8 concerning the number of cookies per page. Profile #8 does not use any extensions but a plain, unmodified browser and hence does not interact with any cookie banner. Hence, we use it as our baseline (reference) for our study, run from the same location (the EU) as the profiles that use the extensions. Note that we do not assume that sites in the baseline profile use less (or more) cookies than sites in other profiles but that we can use the profile to understand how websites use cookies if the user does not interact with the cookie banner.

5.3.2. Effects of the Analyzed Extensions

Table 5.3 provides a high-level overview of the results measured with each profile. Overall, the results hint that the used extensions impact the presented metrics at different magni-

Table 5.3.: Overview of the high-level results per profile. The table lists (1) for each profile the number of third- and first-party cookies; (2) the different cookie categories: "Strictly Necessary" (Strict.), "Performance" (Perf.), "Functionality " (Func.), and "Targeting/Advertising" (Targ.); (3) the number of cookies in the local storage; and (4) number of observed tracking requests.

Nr.	Name	Group	$\sum$ Cookies	1st party c.	3rd party c.	Strict.	Func.	Targ.	Perf.	Local storage	Track Req.
#1	I don't care about cookies	AcceptAll	385 168	157 295	227 833	25 494	19 464	90 369	66 691	115 980	1 977 747
#2	Consent-O-Matic	RejectAll	369 264	155 883	213 346	25 562	18 764	83 438	64 526	113 266	1 891 632
#3	Consent-O-Matic (custom)	AcceptAll	424 089	165 639	258 412	26 917	21 730	100 607	70 371	124 697	2 100 955
#4	NinjaCookie	RejectAll	368 429	155 430	212 961	25 221	18 660	84 869	64 796	112 863	1 917 446
#5	CookieBlock	AcceptFunc	181 351	89 661	91 657	22 865	13 283	26 328	15 060	111 037	2 087 911
#6	SuperAgent	AcceptFunc	416 215	165 268	250 907	26 705	20 771	96 657	69 584	134 821	2 013 957
#7	Custom	AcceptAll	450 204	169 803	280 363	26 773	22 881	108 877	72 957	126 567	2 160 654
#8	None (EU)	Baseline	372 704	152 420	220 242	24 508	18 683	87 248	64 529	108 930	1 918 446
#9	None (US)	Baseline	76 540	183 232	586 267	32 073	34 044	186 613	89 657	152 513	2 542 748

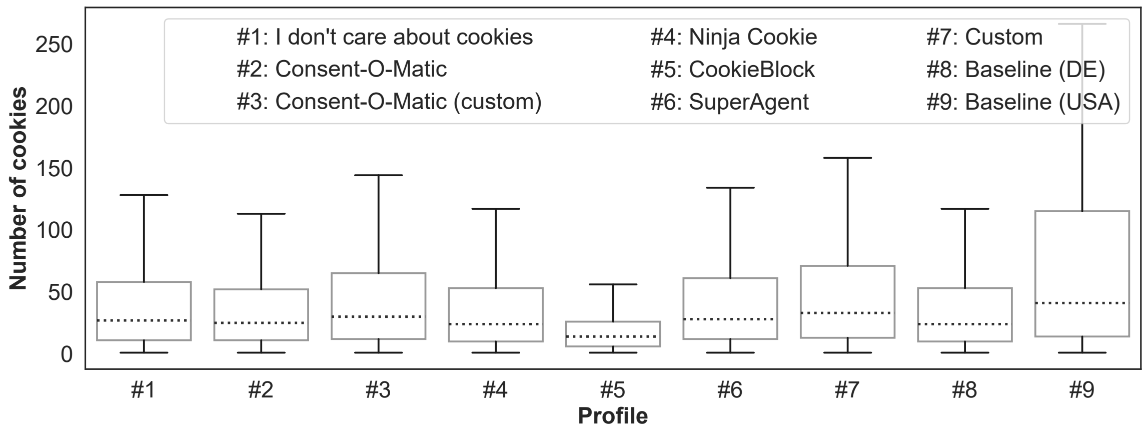

Figure 5.1.: Cookies observed per page with different profiles.

tudes. For example, we record that some essential metrics in terms of privacy (third-party cookies and targeting cookies) vary up to 640%. The average impact on the number of cookies is 83% (SD: 6; min: 0; max: 640). The permutation test found statistically significant (p-value < 0.001) impact of the profile (i.e., extension) on the number of total cookies, first-party cookies, third-party cookies, targeting and advertising cookies, usage of the local storage, tracking requests on page level (p-value < 0.001 for all named metrics). However, for the following metrics, we found *no* statistical impact for "strictly necessary" and "functional" cookies. The results indicate that the impact of the analyzed extensions in our measurement differs notably, which means that depending on the users' requirements, different extensions need to be evaluated. Furthermore, the findings indicate that while all extension aim to fulfill a similar goal (i.e., blocking cookies for users), the rich landscape of different cookie banner implementations makes this an arguably hard and error-prone task. Similar to findings of previous work [161, 34], the measurement performed from the US shows considerably higher values in terms of cookies and tracking requests than the measurements performed from Europe. Furthermore, in the EU baseline profile (#8), the pages used fewer cookies than the profiles in which the extensions accepted (some) cookies and more than those that rejected most cookies. These observations indicate that our measurement approach works as intended, meaning extensions work on some sites and pages). Furthermore, we observe some expected behavior—which also shows that our method works as intended—like increasing tracking requests if we accept all cookies or fewer targeting cookies if we refuse to use them. Based on the observed results, *CookieBlock* is the most effective tool for blocked cookies.

In the following, we analyze the profiles based on the expected functionality of the installed extension (column "group" in Table 5.3) and investigate the effects of these groups separately.

5.3.2.1. Accepting All Cookies

In our setup, we use three profiles (#1, #3, and #7; *AccepAll* in Table 5.3) configured to accept all cookies on a page. Note that for the extension *I don't care about cookies* (#1), it is unclear on which sites and pages it accepts or rejects cookies. Since the extension lists in its description that it primarily accepts cookies—when interacting at all—we include it in the "accept all" group. At first glance, grouping the extension "I don't care about cookies " (#1) might seem counterintuitive since it is not documented how the extension interacts with the banners. However, after manually inspecting the plugin's source code, we have classified it as "Accept all" because the used signatures overwhelmingly accept all cookies if the tool cannot hide the consent banners. To verify this evaluation, we contacted the developer of the extension, who confirmed our observation. As one would expect, the three profiles show an overall average increase of 10% in terms of cookies used by a page. The results show that, on average, a page in profile #1 uses 7 (SD: 11; min: 1; max: 433) cookies, 7.5 (SD: 12; min: 1; max: 433) cookies in profile #3, and 7.9 (SD: 13; min: 1; max: 433) cookies in profile #7. In comparison, in profile #8 (the EU baseline) a page uses 6.8 (SD: 11; min: 1; max: 848) cookies. Overall, the results show that the profiles in this group increase the number of first-party cookies by 4% and third-party cookies by 11% (1#: 4%; 3#: 11%; 7#: 19%;), on average. Hence, the extensions in this group primarily impact third-party cookies.

Based on the numbers, the effect of the extensions seem to be similar. To better understand their impact on the used cookies, we compare the similarity of the cookies set by a page. Overall, a comparison of all profiles in this group with the baseline profile #8, using the Jaccard index (cf. Section 5.2.5), shows a medium similarity of .45 (SD: .46; min: 0; max: 1), on average. The same comparison, when excluding the baseline profile, shows a higher average similarity of .53 (SD: .46; min: 0; max: 1). We get the highest similarity when we test the similarity of #3 and #7 with an average of .64 (SD: .44; min: 0; max: 1). Thus, our analysis shows that even if all three extensions claim to allow similar types of cookies, there are differences in the cookies present in the cookie jar. Therefore, the extensions interact differently with the banners or at least with different success. From a privacy perspective, this finding is concerning since it might be hard for users to evaluate the effect of an extension since sites set a similar amount of cookies, but the used cookies seem different. Counting the occurrence of potentially privacy-harming elements is a metric that ad blockers often use (i.e., blocked trackers) to evaluate effectiveness. Still, there might be better measures for cookie banner tools.

5.3.2.2. Accepting Functionality Cookies

In this section, we analyze the impact of the profiles that accept functional cookies (#5 and #6; *AcceptFunc* in Table 5.3). The two extensions impact the number of cookies a page uses differently. Compared to our baseline, profile #5 reduces the number of cookies per webpage by 40% but the extension used in profile #6 increases the number of cookies per webpage by 8%. On average, pages in profile #5 use 4 (SD: 6; min: 1; max: 433) cookies and pages in profile #6 use 7.2 (SD: 12; min: 1; max: 433) cookies. The notable difference in the effect of the compared tools can be attributed to how they work (cf. Section 5.2.2). *CookieBlock* (#5) uses an ML-approach and *SuperAgent* uses a rule-based approach. The results show that profile #6 increases the number of first-party cookies per page, on average, by 2% and third-party cookies by 7%. In contrast, profile #5 decreases the number of first-party cookies per page by 26% and of third-party cookies per page by 48%, on average. Note that #5 is the only profile showing a notable decrease in first-party and third-party cookies per page.

The distinctive differences between both profiles are also discernible in the similarity of the cookies set by the pages. The Jaccard index shows an average similarity of .48 (SD: .45; min: 0; max: 1) per page. Thus, only roughly half of the set cookies are present in both profiles. Comparing the two profiles with the baseline, the similarity test of the profiles #5, #6, and #8 shows a lower average similarity (mean: .39 (SD: .43; min: 0; max: 1)), which indicates that the extensions affect the used cookies. The results show that some websites seem to wait until a user interacts with a cookie banner before setting specific cookies. Still, approaches that actively delete cookies seem more effective than 'trusting' the page itself.

5.3.2.3. Rejecting All Cookies

Here, we inspect the impact of the extensions that assist users in rejecting all types of cookies (profiles #2 and #4; *RejectAll* in Table 5.3). The results show that, on average, the number of cookies per webpage is affected similarly by both extensions. Compared to the baseline, the number of cookies in profile #2 (6.8 cookies) and profile #4 (6.7 cookies) stays reasonably stable. Within the first- and third-party context, both profiles use a similar number of cookies. The permutation test found no statistically significant (p-value > 0.44) impact by the used extension on the average number of first-party and third-party cookies per page. Furthermore, the bootstrap confidence intervals for profile #2 and #4 are closely aligned, e.g., for third-party cookies we yield [21.1, 22.17] and [21.1, 22.13] for profile #2 and #4, respectively. This observation means that both extensions block the usage of cookies equally , which is expected as they deny a site to set any cookie.

Finally, we test the similarity of the cookies present in both profiles. Overall, we see a similarity of .70 (SD: .43; min: 0; max: 1), on average. The comparison of both profiles with the baseline profile shows a similarity of, on average, .58 (SD: .46; min: 0; max: 1). Thus, even though the number of cookies in the store is similar, the set of used cookies differs. While we also see no high similarity in terms of used cookies for extensions that reject all cookies, we find that this type of extension leads to an overall decrease in used cookies and is similar to our baseline. This finding also highlights that statistically speaking, sites wait until a user chooses how cookies shall be used (the number of cookies remains similar to the baseline).

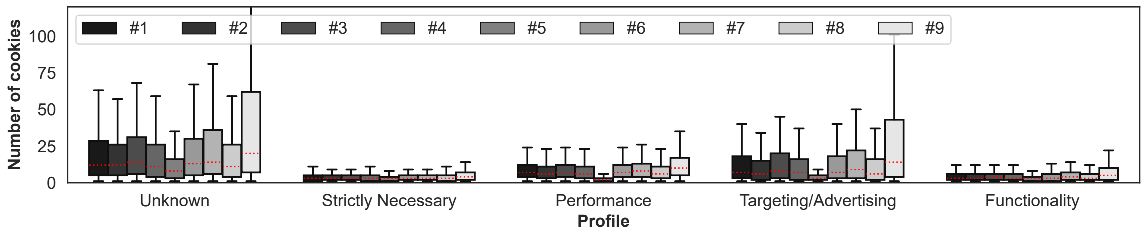

Figure 5.2.: Number of cookies in the different profiles by cookie usage category. To increase readability, the y-axis is cut at 100. The upper whisker of the category "Unknown" in profile #9 is around 150.

Lessons learned

Our observations show that some extensions can limit tracking and data collection by blocking cookies. In contrast, others may increase the number of cookies set by websites, including first-party and third-party cookies. This finding could result in more data collection and potentially negatively impact their privacy. Our findings revealed that the extension *CookieBlock*, in the category of "Accepting Functional Cookies", reduces the number of cookies drastically, while extensions in the "Rejecting All Cookies" category do not have any meaningful effect on the number of cookies. For users who utilize such tools to increase their privacy online, these findings mean that they should rely on a combination of tools (e.g., ad or tracking blockers) rather than only using a cookie banner tool.

5.3.3. Types of Used Cookies

In the previous section, we have shown that the analyzed extensions impact the cookie usage behavior of the investigated sites and pages. In this section, we dive deeper into the observed differences of used cookies to better understand the effectiveness of the extensions. Specifically, we analyze how each profile in our setup affects the cookies based on the cookie's purpose (cf. Section 5.2.5).

5.3.3.1. General Overview of Used Cookie Categories

Previously, we have analyzed in which context the pages set cookies and the impact of the analyzed extensions on them. In the following, we investigate for which purpose cookies are set and the effectiveness of the extensions to block specific types of cookies. Figure 5.2 shows the distribution in the categories of used cookies across the nine profiles. Figure A.5 in Appendix A.4.3 shows this distribution for each analyzed extension. A visual inspection of the figure shows that the extensions impact the used cookies differently. Indeed, the permutation test shows statistical significance (p-value < 0.001) in the used profile and the number of cookies in the categories. Compared to the baseline profile (#8), the results show in all profiles, except #5, that the number of cookies in a category per page increases (*Targeting/Advertising*: 14%; *Performance*: 5%; *Functionality*: 3%; *Strictly Necessary*: 5%). These observations apply to the first- and third-party contexts, but the increases we record are mainly in the third-party context. For instance, our analysis shows an increase of 2% for *Targeting/Advertising* cookies in the first-party context, while these cookies increase, on average, by 17% per page in third-party context. Notably, profile #7 increases the presents of third-party *Targeting/Advertising* cookies, on average by 45%. The results indicate that our method and the analyzed extensions work as intended because the number of cookies in different categories increases in specific profiles only (compared to the baselines).

We record small changes between the profiles regarding *Functionality* and *Strictly Necessary* cookies. However, the permutation test reveals that the number of *Performance* and *Targeting/Advertising* cookies differ statistically significantly (p-value < 0.001) depending on the profile. In the baseline profile, we recorded, on average, 9.2 (SD: 11; min: 1; max: 102) tracking cookies per page. In profile #7, we measured an average increase of 28% in the usage of such cookies, and in profile #5, an decrease of 18% per page, on average. Our observation on *Unknown* cookies shows that the extensions have the same effect on these cookies as on targeting cookies. Thus, the results indicate that each extension impacts the number of cookies in each category differently. Extensions seem to have less impact on cookies that have potentially less impact on the users' privacy (e.g., *Functionality*), which is a positive signal for users who aim to protect themselves against online tracking for different means.

Accepting All Cookies For the extensions that accept all cookies (#1, #3, and #7), we see that the number of *Targeting/Advertising* cookies increases significantly (cf. Section 5.3.1) by 8% on average while the number in the other categories stayed stable (mean increase of 2%), compared to the baseline. Note that the increment mainly occurs in the third-party context. Overall, first- and third-party cookies increase by 4% and 11%, respectively. This (expected) increase in tracking cookies could negatively impact users' privacy as third parties collect and analyze more data about them. To a lesser extent, this increase may result in increased loading times and data usage of pages [124]. Thus, users should weigh the benefits and drawbacks and decide based on their needs.

Accepting Functional Cookies Overall, profile #5 reduces the number of observed cookies in all categories. On average, *Functionality* cookies go down by (9%), *Performance* cookies by (38%), *Targeting/Advertising* cookies by (36%), and the number of *Strictly Necessary* cookies stays stable (0%). The same analysis for profile #6 shows that the number of cookies per webpage stays almost similar. We record an average increases for *Performance* and *Strictly Necessary* cookies by (1%), for *Targeting/Advertising* cookies by (5%), and *Functionality* cookies stays stable (0%). The permutation test reveals a significant difference in the impact of all cookie types, except *Strictly Necessary*, between profiles #5 and #6. The results highlight the importance of carefully considering the privacy implications of different 'cookie banner interaction' extensions, even when they claim to provide similar functionalities. Therefore, it can be challenging for end users to select the extension that meets their privacy needs, as extensions with similar goals and functionalities can have a considerably different impact on users' privacy.

Rejecting All Cookies The profiles that reject all cookies (#2 and #4) have little impact on the number of cookies in each category. Compared to the baseline profile (#8), we observed an increase in *Functionality* and *Strictly Necessary* cookies by 1% and a decrease for *Targeting/Advertising* cookies by 1%. This finding indicates that the extensions have almost no effect on the users' privacy, but they eliminate the need to interact with (annoying) cookie banners). Furthermore, the results indicate that most sites wait until a user chooses before setting cookies.

5.3.3.2. Similarity of Observed Cookies and their Type

In the previous Section, we compared the number of cookies in each category in the different profiles. In the following, we compare, using the Jaccard index, the cookies set by different sites to understand which cookies are kept or removed by the extensions. To allow comparability, we compute the similarity of observed cookies within a functionality group of an extension (cf. Table 5.2). Hence, we only compare profiles where the pages can set the same types of cookies. Figure 5.3 provides an overview of the similarity of the observed cookies in the different categories across the profile groups. "Functional" cookies have high similarity in all groups, including the one that should reject the usage of all cookies. The results of all other types of cookies show differences in the similarity of observed cookies and/or the effect of the different profile groups. Interestingly this also applies to "strictly necessary" cookies, which should be similar across all profiles.

On average, we find a high similarity of .75 across the groups that accept all cookies. However, within the group that only accepts functional cookies, we see an average similarity of .70. The group of "RejectAll" profiles shows an average similarity of .81. Hence, the extensions affect different cookies in these profiles, meaning other cookies are set and used on the analyzed pages. The observed discrepancy in similarity among the extension groups can likely be attributed to their different handling of cookies. On the one hand, the extensions within the "AcceptFunc" group use different approaches to accept cookies with varying success, which is evident by the number of cookies set between profiles #5 and #6. On the other hand, extensions within the "AcceptAll" group tend to trigger more third-party cookies, contributing to a lower similarity.

Targeting/Advertising Cookies Figure 5.3 shows that the installed extensions affect targeting and advertising cookies quite differently. While we see a very high similarity (.83, on average) for extensions that reject all cookies, the similarity of other extension groups decreases. Notably, extensions that allow a more fine-grained control (e.g., group "AcceptFunc") of cookies show a wide range in the similarity of cookies present on a page: .70 (SD: .44; min: 0; max: 1). Hence, the numbers support the claim that the usage of each extension seems to lead to a different set of cookies present in the browser, which makes it harder for users to assess their ability to help them to protect their privacy.

Strictly Necessary Cookies The group of cookies without whom a page would not work correctly shows the most significant deviation (p-value < 0.001) in the similarity of all cookies based on the permutation test. The boxes range from low similarity ($\leq .2$) to perfect similarity (1) for all extension groups. On average, the results show a similarity of .64 for extensions that reject all cookies, .49 for extensions that accept all cookies, and .43 for extensions that only allow functional cookies. The results suggest that extensions have a more noteworthy impact on the similarity of strictly necessary cookies than targeting/advertising cookies. Thus, this disparity highlights the impact that extensions may have on a page's functionality and suggests that there is a need for further investigation of the interaction between the extensions, the (real) purpose of necessary cookies, and the page's functionality.

Lessons learned

Our analysis shows that different extensions for blocking cookies have varying effects on the types of cookies used by the analyzed websites. For example, extensions that accept all cookies tend to increase the number of 'Targeting/Advertising' cookies, while extensions that only accept functional cookies show different results. However, extensions that aim to reject all cookies have a minimal impact on the used cookies and, to some extent, on users' privacy. This finding highlights the challenges users face if they seek to enhance their privacy while browsing the web, as it may take more work to determine the most suitable cookie interaction extension solely based on its stated purpose or description. The results of the similarity measurements suggest that users must carefully consider the functionality and implications of the extensions they choose to install and use them accordingly to balance the trade-off between privacy and functionality.

5.3.4. Observed Cookie Usage Patterns

In this section, we provide an overview of the usage of cookies (e.g., setting or changing a cookie). Our method allows us to distinguish between three operations (1) *added* if a new cookie is set, (2) *changed* if an existing cookie value is changed, and (3) *deleted* if a cookie has been deleted. Figure 5.4 provides an overview of the different cookie access patterns resulting from using the different browser extensions over time. The figure shows the first and third-party cookie "usage patterns" over time after visiting a page. By "usage pattern," we mean how a page uses cookies. For example, when does a page usually set first-party cookies or delete third-party cookies. The different extensions seem not to impact the usage pattern but affect the number of added, changed, or deleted cookies. Thus, in general,

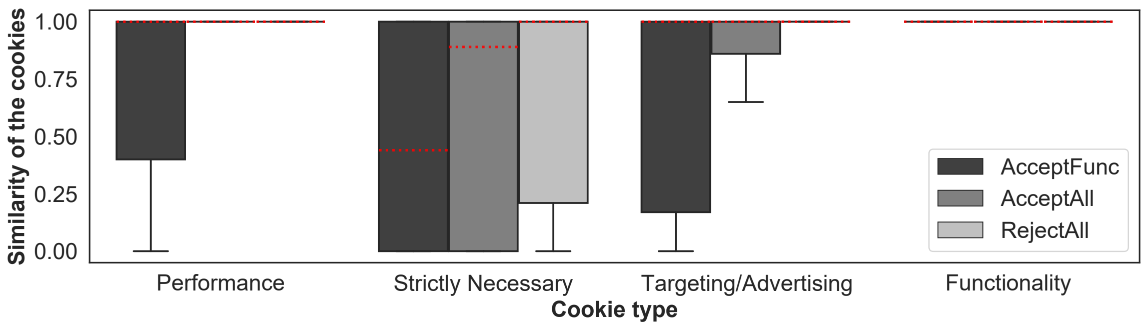

Figure 5.3.: Similarity of the observed cookies for each cookie type by extension group.

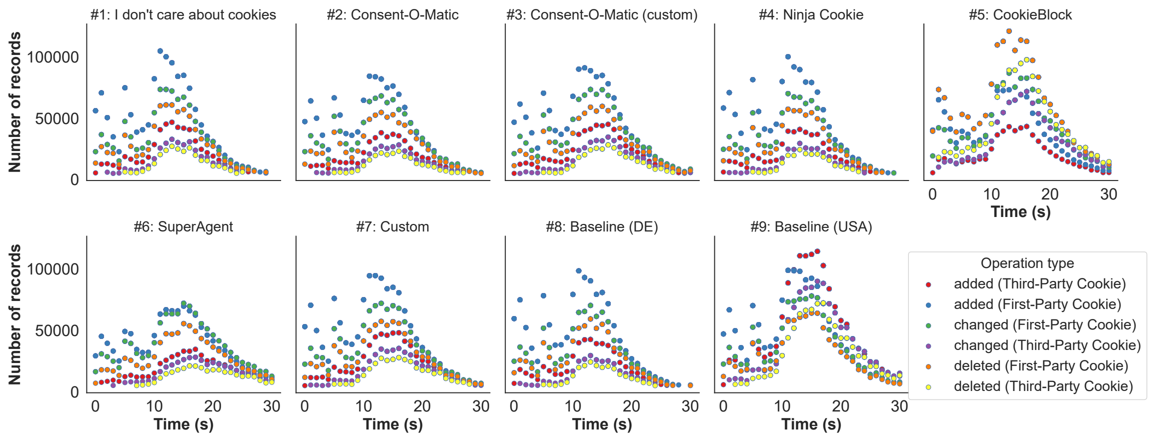

Figure 5.4.: Cookie usage patterns of the analyzed webpages for the different profiles over time.

the users' choice of which cookies might be used does not lead to different strategies for how pages utilize cookies.

Across all profiles, the number of *added* and *changed* cookies rises after the first few seconds after the browser visits the website. Thus, websites wait before setting cookies but use them whether users choose to accept cookies or not. However, by visual inspection, one can observe different cookie access patterns. For example, the number of added and changed cookies drops after their initial setting. After accessing a page, the peak is reached after 13 seconds (SD: 7; min: 0; max: 30) for first-party cookies, and after 15 seconds (SD: 6; min: 0; max: 30) for third-party cookies, on average. On average, operations on third-party cookies are processed 2 seconds later than first-party cookies for each profile. This difference indicates that the analyzed extensions actively lead to more operations on third-party cookies, especially during longer page visits. In profiles #5 (*CookieBlock*) and #9 (US profile), we see a notable increase in deletion operations, in contrast to other profiles.

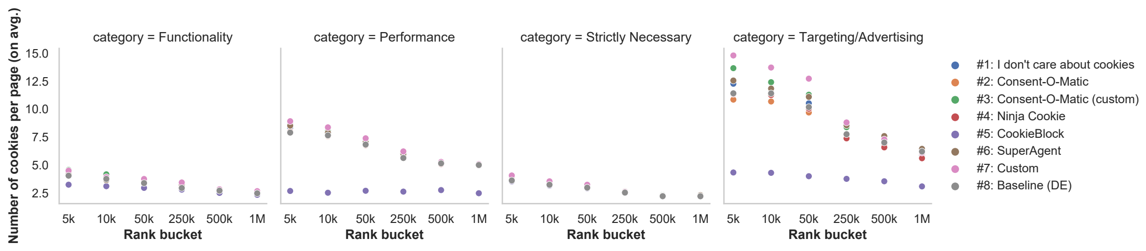

Figure 5.5.: Number of cookies observed in each profile based on the popularity of a site and the type of a cookie.

Lessons learned

Our results indicate that the different extensions impact the access patterns of websites to some extent. However, we did not observe notable differences, meaning that websites do not change their inner workings based on the users' choice to use cookies. Nevertheless, the extensions impact the number of cookies used by each site.

5.3.5. Impact of a Site's Rank

Next, we discuss the impact of a site's rank on the effectiveness of the analyzed tools. More precisely, we analyze if the tools work better on popular (e.g., because they tend to use more CMPs) websites than on less popular sites. Figure 5.5 provides an overview of the relation between the purpose of a cookie and the visited sites' rank for the analyzed extensions. Overall, we see that popular websites use more cookies, which aligns with previous work, but also that the analyzed tools are more effective on these websites. Furthermore, the differences in the effectiveness of the analyzed extensions are more considerable for top sites. Thus, less popular sites seem to have a more diverse set of cookie banners, making it more challenging for extensions to interact with them than popular sites. The extension *CookieBlock* (profile #5) outperforms all other extensions across all cookie categories and ranks of sites. For all cookie types, we see a steady decrease in the number of used cookies, and the results show that the delta between the most effective extension (in terms of cookies) and the least effective extension also shrinks. For example, for targeting cookies on popular websites (rank 1–5k), the absolute difference in the number of set cookies is 4 (SD: 1.4; min: 11; max: 15), but for the least popular sites (rank 500,001–1M) the delta is only .8 cookies (SD: .28; min: 5.6; max: 6.4). These figures exclude profile #5 due to its meaningfully better performance.

Lessons learned

Our results indicate that the performance of cookie-blocking extensions may vary depending on the popularity of the visited website. Popular websites tend to use more cookies, and the differences in the effectiveness of the analyzed extensions are more pronounced for these sites. The best-performing extension in terms of blocking cookies, *CookieBlock*, demonstrates a clear advantage across all cookie categories and website ranks. However, it should be noted that the difference in the number of cookies set between the best and least effective extensions decreases for less popular websites. This observation suggests that users may experience varying levels of cookie blocking depending on the websites they visit, with potentially more diverse effects on popular websites.

5.3.6. Cookies in the Local Storage

The *HTML Web Storage API* ("local storage") is a way to store data on the client. Compared to classic HTTP cookies, the storage can hold large amounts of data without negatively impacting a page's performance. It works similarly to HTTP cookies as objects in the local storage are key-value pairs that can be accessed via JavaScript [109]. In the following, we use the term *object* rather than *cookie* for objects stored in the local storage to make it easier to distinguish between HTTP cookies and data in the local storage. On average, we identified 122k objects in the local storage of the profiles. A site uses 12 (SD: 31; min: 1; max: 2,720) keys on average, which is 60% less than the average number of cookies.

Here, we analyze the differences the profiles cause in the elements stored in the local storage. Again, we use profile #8 as our baseline. On average, a site in profile #8 uses 11.6 objects in the local storage (cookies). The permutation test shows significant (p-value < 0.001) differences between the profiles that accept all cookies (#3 and #7) and the baseline. All other profiles do not significantly impact the number of local storage objects per site (p-value ≥ 0.62). We record that the profiles #3 and #7 increase the number of keys per site by 13% (1.5 keys), on average. The increase could be because some sites may use the local storage to store more data than they could store in HTTP cookies, as the local storage allows for larger amounts of data. Furthermore, websites may use the local storage in combination with HTTP cookies to store data more persistently, as the data stored in local storage has a longer lifespan than cookies.

Profile #6 is the only profile in which fewer objects are present in the local storage (21%). We tried to use the classifications provided by *Cookiepedia* to classify the usage purpose of objects in the local storage. However, only 6% of the keys could be classified. Due to this low success rate and the low impact of extensions on this type of storage, we decided against a deeper analysis.

Lessons learned

The results show that the use of local storage objects, or key-value pairs stored on the client, is not significantly impacted by the extensions, except for profiles that accept all cookies. These profiles increased the number of objects stored in the local storage.

5.3.7. Potential Subsequent Effect

This section analyzes the potential impact of different extensions on tracking mechanisms. Overall, on average, interacting with consent banners increases the number of HTTP requests per page by 5%. The baseline profile (#8) has, on average, 89 (SD: 113; min: 1; max: 18,208) HTTP requests and 15.5 (SD: 57; min: 1; max: 18,154) tracking requests per page. Here, we dive deeper and test each extension group's impact separately to understand their effects.

Accepting All Cookies Overall, the results show that profiles #3 and #7 notably increase the number of tracking requests. In profile #3, we see an average increase of tracking requests per page by 12%, for profile #7 by 8%, and for profile #1 by only 3%. The permutation test found a statistical significance (p-value < 0.001) impact of these extensions on the number of tracking requests (cf. Section 5.2.5). Let's only look at profile #7, for which we can measure the success rate of interaction with cookie banners. We find that the average number of tracking requests per page increases by (60%) (from 16.6 to 26) after interacting with the cookie banner, and the bootstrap confidence intervals for both profiles differ: we yield [17.12, 17.39] and [15.29, 15.90] for profile #7 and #8, respectively. The results demonstrate that accepting cookies leads to increased use of cookies and other privacy-harming techniques—in this case, user tracking. Thus, the used extensions can have a far-reaching impact on users' privacy than 'just' blocking cookies.

Accepting Functionality Cookies The observations of the group *AcceptFunc* show different results for both profiles. We see that profile #5 increases the average number of tracking requests per page by 8%, while we record a minimal increase of 3% for profile #6. The bootstrap confidence intervals for the profiles differ with [16.69, 16.96] and [15.86, 16.10] for profile #5 and #6, respectively. Note that profile #5 (*CookieBlock*) deletes cookies rather than telling websites not to use them. Thus, these finding indicates that extensions with the same purpose can have vastly different impacts on the users' privacy, as different handling of cookies can lead to unexpected results, such as an increase in tracking requests. This result challenges the assumption that blocking cookies (especially targeting ones) will always lead to fewer tracking requests and highlights the need for further investigation of this phenomenon.

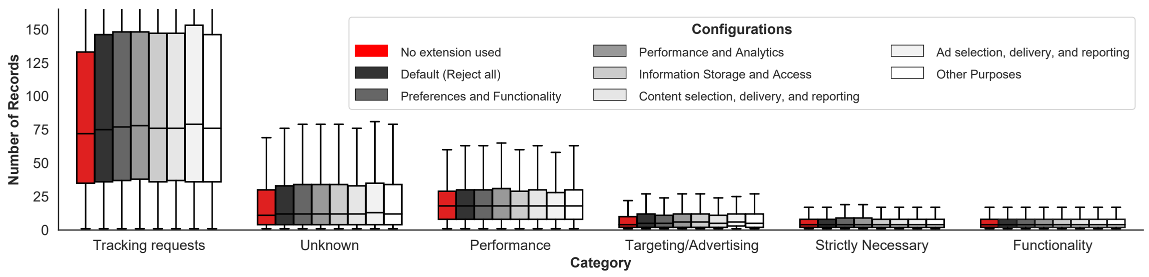

Figure 5.6.: Distribution of number of records for tracking requests and different cookie categories grouped by different configurations (allowed cookie types) of *Consent-O-Matic*.

Rejecting All Cookies We turn our analysis into the effects of the group *RejectAll*. Overall, we see that the profiles in this group do not affect the tracking requests on the pages, and the number of tracking requests per page stays relatively stable (mean of the profile #2: 15.4; #4: 15.5) in the observed profiles. The bootstrap confidence intervals for both profiles stay similar and yield [15.25, 15.50] and [15.29, 15.70] for profile #2 and #4, respectively. In this case, extensions that reject all cookies do not negatively impact the number of tracking requests, making them a suitable option for end users interested in reducing tracking on their devices.

Lessons learned

It is important to note that each extension may affect privacy, functionality, and performance differently. For example, profile #5 substantially decreases the number of targeting cookies but increases the number of tracking requests per page. This finding highlights the challenge of balancing privacy and functionality for users. Additionally, the results show that different extensions, even with similar goals and functionalities, can have noteworthy differences in their impact on tracking mechanisms. For instance, profiles #5 and #6, in the *AcceptFunc* group, have different effects on the number of tracking requests per page, despite being designed to accept only functionality cookies. To conclude, the results demonstrate the complexities of balancing privacy and functionality for users and the potential consequences of using such extensions.

5.3.8. Different Extension Configurations

Some of the examined extensions offer different configuration options of which types of cookies the extension should allow or deny if the visited site provides a choice (cf. Table 5.2). To understand the effect on users' privacy of these options, we measure the effectiveness of the different configurations of the most popular extension in our corpus that offers multiple configuration options: *Consent-O-Matic*. In particular, we performed separate measurements using eight different extension configurations. Note that we measure each available option individually but in combination with other options and only analyze the top 7,000 sites from our dataset (cf. Section 5.2.1). Appendix A.4.4 describes the used setup in more detail.

Figure 5.6 provides an overview of the results of the comparison between the different options (grayscale bars) and a profile with no extension installed (red bar). While the extension offers many different configuration options, each configuration's outcome (effect) is similar. Note that the default configuration rejects all cookies. Within each cookie category, the results are comparable with the most variations for '*Tracking/Advertising*' cookies. Note that the extension always accepts '*Functionality*' cookies no matter which configuration is used, which explains this category's almost identical effect size. Table A.5 further provides an overview of the cross-comparison between the configurations. The experiment's results show that the different configuration options have similar effects on the tracking requests and number of cookie types, with minor variations. This finding suggests that the granular configuration options do not considerably enhance their privacy control in the current state of the extension. It is important to note that the analyzed extension is specialized in interacting with CMPs and, therefore, only has a limited effect on sites that do not use them. However, our findings indicate that due to missing standards on how to interact with cookie banners and banners that do not offer fine-grained options to limit the usage of cookies, offering multiple options to users does not help them, at least in our experiment. However, this might change in the future if consent mechanisms get more elaborate and automatically interacting with them gets easier.

5.3.9. Repeated Experiments

Measuring the Web is challenging due to its dynamic nature. Previous work has shown that repeated measurements are essential to assess the dynamic effects of a measurement [34] and that the vantage point [87] of a measurement can considerably affect an experiment's outcome. To verify the correctness of our results, we performed three temporally close measurements from different vantage points from within the EU (cf. Appendix A.4.6) using *Amazon Web Services*. We use the same profiles as in our primary measurement (cf. Section 5.2.2) in all three measurements. However, we reduce the scale of each measurement for simplicity. On average, we visit 612,000 pages on 7,000 sites for each profile. We choose the top 7,000 sites from the used Tranco list and randomly sample 15 pages from each of the used buckets (cf. Section 5.2.1).

For this experiment, we resort to the volume and type of observed cookies and tracking requests measured in each profile and compare the three measurements based on these fundamental data points. The rationale is that if we find statistically significant overlaps in these numbers across the measurements, one can assume that our overall results are correct. We find that the analyzed pages set a similar amount of cookies in the used profiles across the three measurements: on average, we observe a delta of 2.2 (SD: 11; min: 767; max: 324) cookies on the analyzed pages. The PERMANOVA test found no statistical significance (p-value > 0.87) neither between the measurement runs (i.e., country and time) nor the number of cookies set on each page in each profile. However, the Kruskal-Wallis test shows varying results for some combinations of profiles on specific metrics, such as tracking requests and types of cookies. For instance, five profiles show significant differences in targeting cookies (p-value <0.001). Given these varying results, we investigate the magnitude of these differences by measuring the effect size. The effect size provides additional information beyond statistical significance, measuring the degree of difference. In this study, we use the Eta-squared (η^2) measure to assess the effect size, reflecting the proportion of total variance attributable to an effect. Our results show that all computed η^2 values are less than 0.01 (SD:.0003; min: .00001; max:.001), suggesting that while some differences are statistically significant, their practical impact is minimal. We attribute these findings to the Web's dynamic nature and argue that while some differences between measurement runs exist, they do not tamper with the reliability or correctness of the results.

5.4. Recommendations

Based on the results, we provide actionable recommendations for researchers, developers, and users of "cookie banner interaction" tools.

5.4.1. Recommendations for Researchers

Among many other things, researchers might want to utilize "cookie banner interaction" tools if they are interested in how websites use cookies or to simulate a more 'realistic behavior' of a site in a Web measurement. However, our results suggest that the usage of such tools creates another, to some extent uncontrollable, bias as the effectiveness of the tools varies, and it is (without manual analysis) not possible to assess on which pages the tools could interact with a banner and on which they could not. Therefore, researchers must carefully consider whether (or to what extent) such a tool is tenable.

Recommendation #1: ✓

When planning a study, researchers should evaluate on which pages the utilized tools work (e.g., by instrumenting them) to assess the impact on the outcome of the study.

Finally, researchers use the tools to reduce limitations affecting Web measurements (conducted from the EU). However, our results show that the tools would impact a study's outcome differently, even if configured to accept the same categories of cookies, which hinders the comparability of different studies. Thus, it is inevitable to contemplate the effect a tool might have on a study.

Recommendation #2: ✓

Researchers need to identify new limitations and challenges that arise when utilizing a cookie banner tool and should rigorously weigh the benefits and drawbacks for the study.

5.4.2. Recommendations for Developers

Developers develop the extensions to (1) maximize the coverage of the tool (i.e., number of banners and websites to interact with) and (2) to implement the user's choice as well as possible. To maximize their coverage, extensions should not only resort to interacting with popular CMPs but extend their scope. It is challenging for developers to keep up with the numerous implementations of different cookie banners. Similar to detecting trackers and ads, a community-driven approach to identify banners and choices within them could help handle this scaling issue. A promising attempt in this direction is the *EasyList Cookie List* [41], which can be used as a starting point to identify banners. Furthermore, our results suggest that the analyzed extensions perform notably worse on less popular sites, which means that such sites should be included when testing an extension.

Recommendation #3: ✓

Developers can include community-driven projects helping identify banners to maximize the coverage of the tool. Also, the testing of applications should incorporate less popular sites.

According to the results, the only tool that actively checked for a cookie's purpose (*CookieBlock)* showed the best performance regarding the number of present cookies. However, such classification of cookies is challenging. Again, community-driven projects like the named *EasyList Cookie List* can assist developers in this regard.

Recommendation #4: ✓

Developers may consider actively checking for (and deleting) cookies that contradict the user's privacy configuration to increase a tool's effectiveness and, to some extent, its coverage.

5.4.3. Recommendations for Users

Users might install "cookie banner interaction tools" for two reasons: (1) they do not want to be bothered by cookie banners, or (2) they want to automate the process to communicate their cookie preferences. Users that use the extensions to improve their privacy online should choose an extension that limits the usage of cookies that may impact their privacy (e.g., "tracking cookies") and should *not* use an extension or configuration that accepts all cookies. At the time of writing this thesis, the extension *CookieBlock* showed the best performance regarding blocking (deleting) unwanted cookies. Finally, the results suggest that other tracking activities may increase if cookie usage is limited. Thus, privacy-aware users need to consider using additional tools (e.g., *Privacy Badger* [156], *Ghostery* [54], or *uBlock Origin* [65]) that help them to limit other forms of Web tracking.

Recommendation #5: ✓

Users of cookie banner interaction tools should additionally use ad or tracking blockers to protect their privacy and should not use configurations that accept all cookies.

Additionally, users must be cautious when choosing a tool regarding its coverage. When choosing a tool, users should generally understand how a tool works and which limitations come from specific design choices. For example, our analysis shows that popular tools focus on interacting with popular CMPs, meaning such tools will only work on sites that utilize these CMPs. Other tools aim to hide banners, which might tamper with a site's functionality as users cannot allow specific cookies; thus, some functions might not work correctly. Nevertheless, in our experiment, none of the analyzed extensions broke a considerable number of sites.

Recommendation #6: ✓

Depending on their goals, users should check how an extension works (i.e., hiding a banner vs. interacting with them) and if the handling method meets their expectations.

5.5. Limitations and Ethics

Our study comes with several limitations. The main limitation of our work is that we cannot measure that the analyzed extensions work as they claim they do. More specifically, we cannot measure if a website ignores a user's choice (or an extension's choice on the user's behalf) or if an extension is not working correctly. Our experiment can only report different tools' effects on the cookies in the browser's cookie jar. However, our study aims to understand the effect of these extensions on users' privacy, which means that the execution of the user's choice is out of the scope of this work and should be explored in future work. Furthermore, our computation of the similarity of set cookies only compares the presents of cookies based on their keys and not their functionality. A page could set a user-specific key, but this key could be used for the same purpose across all users. Finally, since our crawler does not interact with the pages as a regular user would, we only observe an excerpt of a page's functionality. Therefore, our results can only be seen as a lower bound and incomplete. However, implementing a system that automatically generically interacts with any given page to trigger a different state is probably impossible and is out of the scope of this work. Our approach follows best practices and uses well-established tools to reduce the effect of the named limitations.

Our study uses *OpenWPM* to visit the sites and pages in our dataset. However, the way the framework instruments the browser is detectable by a website so that the website might serve different content if and when *OpenWPM* is detected [94]. We have not performed any means to circumvent such detection mechanisms as this might be ethically questionable. Furthermore, a site might detect that we use the *Google Cloud Platform* and *Amazon Web Services* for visiting the pages and serves different or no content [78]. We filter out sites and pages that we cannot reasonably compare to contain the effects of such crawling detection mechanisms. Our filter policy excludes a site if it does not serve a page for at least one crawler, meaning the crawler has been detected. If the site detects all our crawlers but acts consistently across all crawlers (e.g., showing a blank page), our filter does not take effect. Note that this does not change the overall findings of our work notably: When a blank site is served, no cookie banners are present, and no cookies are used. In this case, all extensions have the same effect.

Since our study excludes human subjects, it was exempt from the Institutional Review Board's review. However, like all web measurement studies [34, 87, 161, 162], our study has some inevitable ethical considerations. By automatically visiting several pages, we create traffic and use up resources. We limit the effects by only visiting up to 15 website pages. Also, our crawlers may get served ads, increasing the difference in served and viewable impressions and thus potentially decreasing the revenue of the advertiser.

5.6. Related Work

The area of cookie banners or cookie notices and their impact on users' privacy has been widely researched in recent years. Several works look at the structure and usability of such tools and investigate the impact of various factors, such as position, choice type, and content framing, on users' interactions with cookie consent (e.g., [163, 33]). In contrast other studies [e.g., 66] aim to understand the impact of privacy laws by mapping the formation of the consent management provider ecosystem and measuring its adoption. Studies like [52] investigate the legal compliance of the consent banners, while further works study the use of dark patterns when designing them [159, 63]. Our work distinguishes from these research streams as we investigate the effect of browser extensions that interact with cookie banners automatically or, more specifically, the impact these extensions have on users' privacy.

Related to our research are works that look at the functionality of such banners on a technical level. [108] analyzed legal aspects in the storage of consent through cookie banners by crawling websites, and they found potential legal violations in the storage of user consent. [138] manually analyzed the effect of cookie notices and found that such banners often did not work as intended. Different works look at compliance issues with existing consent management tools [123, 92, 18] and find that websites often do not honor the users' choices.

Other works focus on assisting users by automatically interacting with cookie banners. [119] present *Consent-O-Matic*, a tool that can interact with different cookie banners. Most recently, [18] present a machine-learning-based approach to classify cookies and to delete cookies of specific categories that the user can select. [71] also introduce a machine learning-based approach to classifying the purpose of a cookie. [91] conduct a large-scale crawl to explore the security impact of consenting to a cookie banner and find that users who consent to Web tracking are exposed to more security-sensitive data flows and are vulnerable to more client-side cross-site scripting (XSS) vulnerabilities. [84] present a measurement focusing on popular websites in Europe and the US to study the impact of privacy banners on the web. The authors developed a Web crawler, which can accept privacy policies, and compared the changes in websites before and after accepting the policies.

In contrast to the named works, this work aims to understand the effect of different "cookie banner interaction" browser extensions on users' privacy. By analyzing the functionalities of these extensions, we aim to provide insights into the effectiveness and impact of these tools in protecting users' privacy.

5.7. Discussion and Conclusion

This study compares six different tools that assist users in interacting with cookie banners in a large-scale web measurement study. We configured the extensions to communicate different types of cookies that the users are willing to accept or reject. The results show that the effectiveness of the different tools, in terms of cookies set by a visited page, varies significantly. We find differences in the number of used cookies, type of used cookies, and set cookies (i.e., different keys), even for similar configured tools. Our findings reveal that if users find consent banners disruptive while browsing, they can use the extensions tested in this study to eliminate them with varying degrees of success. However, it is essential to note that extensions with similar purposes lead to different sets of cookies present on a page. For example, *Consent-O-Matic* (#2 and #3) only works with so-called "Consent Management Platforms" (CMPs) and may not help block other types of consent banners.

Furthermore, the results suggest that blocking targeting cookies does not necessarily result in improving user privacy, as it can lead to an increase in tracking requests. This observation could tamper with the intention of the users to counter Web tracking attempts by sites. Therefore, end users seeking to protect their privacy—through cookie banner blocking extensions—should consider the trade-offs between functionality and privacy and select the tool that best fits their needs. However, further research must clarify the correlation between tracking requests and blocking tracking cookies.

In summary, the analyzed extensions aim to provide a user-friendly way to control the cookies users want to accept or reject. However, this study shows that the effectiveness of the extensions varies and that it can be challenging for users to assess or compare the extensions' impact on their privacy. Our findings show that, while these tools grow in popularity, they can be helpful for users to give or withdraw consent for the usage of different cookie types.

Chapter 6

Conclusion and Outlook

In this final Chapter, we summarize the outcomes of our investigation of the robustness of Web measurements (RQ1), their application to Web security (RQ2), and Web privacy (RQ3) in Section 6.1. We then explore potential future research directions in Section 6.2, and finally provide an overall conclusion of this thesis in Section 6.3.

6.1. Research Outcomes

The approach of this thesis consists of three parts: Exploring the robustness of Web measurements, identifying the challenges and possible solutions (RQ1), and ways to enhance the robustness of Web security (RQ2) and Web privacy (RQ3) measurements based on these insights. The findings obtained with respect to the questions formulated are summarized below.

RQ1 Robustness ?

How can we ensure the robustness of Web measurement studies, and which factors lead to variances in security- and privacy-related Web measurements?

In Chapter 3, we investigated the impact of the measurement environment on the results. We conducted several measurements in different environments (see Section 3.1.2) and analyzed the effects in detail. We demonstrated how even minimal design decisions (e.g., browser type, location, or interaction on the website) can significantly influence the results and how they affect the robustness of studies. We published an open-source Framework *MultiCrawl*[1] to conduct comprehensive Web measurements. To enhance the robustness of Web measurements, we developed a criteria catalog (see Table 3.1) for different study phases and provided researchers with guidelines.

To better understand the complexity of the Web, we analyzed the factors that affect Web measurements. To this end, we represented websites as trees that model all resources and their dependencies in another comprehensive measurement (see Section 3.2). We compared the trees of the websites based on different profiles and precisely identified factors that lead to variances in Web measurements. We specifically highlighted that the type of resource

[1] `https://github.com/nrllh/multicrawl`

in a webpage (e.g., iframes or images) and the context in which it loads (e.g., first-party or third-party) play crucial roles in causing variations between Web measurements. In both studies, we show how minimal design choices can lead to incomparable results.

RQ2 Security ?

Having the findings from RQ1, how can we conduct Web security measurements, as a case study to understand the update behavior and vulnerability of websites?

In Chapter 4, we demonstrated in a case study how our findings from *RQ1* can be applied to enhance Web security. We conducted an analysis of an open-source dataset (*HTTPArchive*) to understand how website operators keep their Web environments up-to-date and the potential implications of using outdated software on the security of these environments. We found that 96% of the components used in the global Web are outdated, most of which are over four years old on the average. We also identified that 95% of all analyzed websites use at least one software product for which vulnerabilities have been publicly reported. Based on our findings, we provide actionable recommendations for researchers (e.g., utilizing open-source dataset), developers (e.g., enabling auto-updates and adopting *Semantic Versioning*), and end users (e.g., being aware of website vulnerabilities and using modern web browsers). In this study, we found that using an open-source dataset for large-scale measurements is the only way to ensure the complete reproducibility of a study (taking into account our criteria catalog).

RQ3 Privacy ?

Having the findings from RQ1, how can we conduct Web privacy measurements, as a case study to understand the effectiveness and impact of privacy tools?

In Chapter 5, we demonstrated how our findings from *RQ1* can be applied to enhance Web privacy study. We analyzed six privacy tools interacting with website consent banners. We showed that these tools have significantly different effects, even those offering the same functions. We also demonstrated that such tools can lead to unexpected impacts, such as increased tracking requests. We finally provided actionable recommendations for researchers (e.g., evaluate the effectiveness and the impact of tools), developers (e.g., incorporate community-driven projects, maximize the coverage of tools), and end users (e.g., use ad blockers, consider the trade-offs between functionality and privacy).

Our measurement data, code, evaluation, and additional artifacts for all studies of this thesis are publicly available in Appendix A.1.

6.2. Future Work

In this thesis, we analyzed the robustness of Web measurements and explored how their robustness can be enhanced. We also examined how our findings can be applied to Web security and privacy studies. The following sections will discuss the future works in different domains.

6.2.1. Robustness of Web Measurements

Our results indicate that ensuring complete reproducibility or replicability in Web measurements is a challenging task influenced by various factors. We conducted measurements with 74 different profiles and found that complete reproducibility and replicability can be achieved by utilizing open-source databases (e.g., *HTTPArchive*) while considering our criteria catalog (Table 3.1). For instance, we showed that even crawling a webpage simultaneously with two identical measurement profiles yields different results. These findings suggest that, as a community, we need metrics (e.g., error rate or variance of the measurement) to enhance the robustness of Web measurements. While such metrics are standard in other disciplines, they have not yet been introduced in Web measurements to the best of our knowledge.

Our observations also show that compared to other disciplines, such as medicine or psychology, more research into *Metascience* is needed computer science. The computer science community currently attempts to increase studies' reproducibility and replicability by introducing badges and artifact committees. However, we should invest more resources in critically examining processes during the research cycles and develop strategies, frameworks, and additional guidelines to achieve more robust study results, as demonstrated in Chapter 3 and "Guidelines for Performing Safe Measurement on the Internet" [101].

The following potential questions arise for future work:

- Which metrics can be developed to assess the error rate and variance of observed parameters in Web measurements?
- How can interest in and use of *Metascience* be fostered to critically examine research in the field of computer science?

6.2.2. Web Security

Our case study shows that almost all websites (95%) use at least one software component for which vulnerabilities have been reported and often utilize very old software components. We also showed that the update process usually is not automated and website operators do not consistently process updates. While there may be justifiable reasons for not immediately processing updates (e.g., compatibility issues or end-of-life software), the consequences remain significant (e.g., increased vulnerability). While standards like

ISO/IEC 27002 [149] are often very organizational, there are currently no standards that systematically address the issue of software updates and offer solutions, specially for developers to the best of our knowledge.

In recent years, there have been some developments at the political level. The *European Union's Ecodesign for Sustainable Products Regulation* [48] aims to reduce electronic waste and obligate manufacturers to more efficiently provide security updates for software components for more extended periods. However, a significant part of the software components on the Web is open-source, and such regulations and standards have little impact on these components, making their practical impact on Web components very inefficient. There is a need to develop approaches tailored to Web software components to increase the efficiency of the update process for these components. More specifically, we see a need for approaches that differ e.g., for JavaScript components and content management systems due to these components' static and dynamic nature. In this way, factors like compatibility issues can also be addressed.

The following potential question arises for future work:

- Which open-source approaches can be developed to increase the efficiency of update processing for software components on the Web?

6.2.3. Web Privacy

Our case study on Web privacy has shown that tools intended to enhance user privacy exhibit significant variations and can have unintended side effects that may compromise user privacy. We further observed varied outcomes for tools with identical functions, such as rejecting all cookies. Moreover, we found that users often need to be aware of these discrepancies. Our contact with developers of these tools revealed that even they were unaware of these side effects. For this study, we provided actionable recommendations for the relevant stakeholders. However, various domains still need to be addressed.

Our findings have raised new challenges and questions that concern researchers, developers, and end users and should be addressed. From our point of view, developers of such tools bear significant responsibility. They should not only increase the efficiency of these tools and protect user privacy but also examine the side effects of these tools. Users of these tools (e.g., end users and researchers) also need more reliable information on the efficiency and side effects of these tools to make informed decisions.

The following potential question arises for future work:

- How can the efficiency of privacy tools be enhanced and their side effects addressed during the development phase?

6.3. Concluding Thoughts

Technological advancements are accelerating and becoming more complex, increasing the methodological complexity of our studies. Therefore, the robustness of studies, especially on the Web, remains of utmost importance. Our exploration of Web security and privacy studies has revealed the significance of maintaining rigorous and reliable experimental setups and measurements. As the landscape of the Web is evolving continuously, so do the challenges and considerations associated with measuring its various facets.

The three research questions that drove this thesis provided a holistic view of the current state of Web measurement studies' robustness while shedding light on the intricacies of Web security and privacy aspects. The implications of our findings reach beyond academia and touch on broader societal concerns through actionable recommendations.

From understanding the factors influencing Web measurements to Web security and privacy, our journey paints a comprehensive picture of the current state of affairs. Yet, as all rigorous scientific inquiries reveal, every answered question uncovers further questions. The need for continued research, innovation, and, most importantly, collaboration across disciplines (e.g., *Metascience*) becomes evident.

Beyond the technical facets and analytical perspectives, a powerful message is obtained: The quest for a more secure and private Web experience is inextricably linked to the robustness of our measurements. Without robust measurements, our understanding remains fragmented, and our interventions are potentially misguided.

As we are advancing, it is essential to continually reassess our methodologies, champion open and collaborative research, and prioritize the broader ethical and societal implications of our work. This thesis, we hope, will not only serve as a culmination of our endeavors but also as a beacon to guide future explorations in the domains of Web measurements, security, privacy, and beyond — for a reliable, safer, and better tomorrow for all of us.

Bibliography

[1] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. "The Web Never Forgets: Persistent Tracking Mechanisms in the Wild". In: *ACM Conference on Computer and Communications Security*. CCS. 2014. DOI: `10.1145/2660267.2660347`.

[2] Dominik Adler, Nurullah Demir, and Norbert Pohlmann. "Angriffe auf die Künstliche Intelligenz – Bedrohungen und Schutzmaßnahmen". In: *IT-Sicherheit – Mittelstandsmagazin für Informationssicherheit und Datenschutz* (Jan. 2023).

[3] Bernhard Ager, Wolfgang Mühlbauer, Georgios Smaragdakis, and Steve Uhlig. "Comparing DNS Resolvers in the Wild". In: *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. IMC '10. Melbourne, Australia: Association for Computing Machinery, 2010. ISBN: 9781450304832. DOI: `10.1145/1879141.1879144`.

[4] 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: *International Conference on World Wide Web*. TheWebConf. 2020. DOI: `10.1145/3366423.3380113`.

[5] Alexa Internet, Inc. *The Top 500 Sites on the Web*. `https://web.archive.org/web/20210715094515/https://www.alexa.com/`. Online; Accessed: 2021-07-15. 2021.

[6] Mark Alllman and Vern Paxson. "Issues and Etiquette Concerning Use of Shared Measurement Data". In: *ACM SIGCOMM Internet Measurement Conference*. IMC. 2007. DOI: `10.1145/1298306.1298327`.

[7] Marti J Anderson. "A new method for non-parametric multivariate analysis of variance". In: *Austral ecology* 26.1 (2001). DOI: `10.1111/j.1442-9993.2001.01070.pp.x`.

[8] Waqar Aqeel, Balakrishnan Chandrasekaran, Anja Feldmann, and Bruce M. Maggs. "On Landing and Internal Web Pages: The Strange Case of Jekyll and Hyde in Web Performance Measurement". In: *ACM SIGCOMM Internet Measurement Conference*. IMC. 2020. DOI: `10.1145/3419394.3423626`.

[9] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. "Dos and Don'ts of Machine Learning in Computer Security". In: *USENIX Security Symposium*. Usenix Sec. 2022.

[10] Association for Computing Machinery. *Artifact Review and Badging Version 1.1.* `https://www.acm.org/publications/policies/artifact-review-and-badging-current`. 2020.

[11] Nimrod Aviram, Sebastian Schinzel, Juraj Somorovsky, Nadia Heninger, Maik Dankel, Jens Steube, Luke Valenta, David Adrian, J Alex Halderman, Viktor Dukhovni, Emilia Käsper, Shaanan Cohney, Susanne Engels, Christof Paar, and Yuval Shavitt. "DROWN: Breaking TLS Using SSLv2". In: *USENIX Security Symposium.* USENIX Sec. 2016.

[12] Keith A. Baggerly and Kevin R. Coombes. "Deriving chemosensitivity from cell lines: Forensic bioinformatics and reproducible research in high-throughput biology". In: *The Annals of Applied Statistics* 3.4 (2009). DOI: `10.1214/09-AOAS291`.

[13] Amitav Banerjee, UB Chitnis, SL Jadhav, JS Bhawalkar, and S Chaudhury. "Hypothesis testing, type I and type II errors". In: *Industrial psychiatry journal* 18.2 (2009). DOI: `10.4103/0972-6748.62274`.

[14] Adam Barth. *HTTP State Management Mechanism.* RFC 6265. Internet Engineering Task Force, Apr. 2011.

[15] Anders Barth. *The Web Origin Concept.* RFC 6465. Internet Engineering Task Force, 2011.

[16] Adrian Bermudez-Villalva, Mirco Musolesi, and Gianluca Stringhini. "A Measurement Study on the Advertisements Displayed to Web Users Coming from the Regular Web and from Tor". In: *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW).* 2020. DOI: `10.1109/EuroSPW51379.2020.00072`.

[17] Sarah Bird, Ilana Segall, and Martin Lopatka. "Replication: Why We Still Can't Browse in Peace: On the Uniqueness and Reidentifiability of Web Browsing Histories". In: *Symposium on Usable Privacy and Security.* SOUPS. 2020.

[18] Dino Bollinger, Karel Kubicek, Carlos Cotrini, and David Basin. "Automating Cookie Consent and GDPR Violation Detection". In: *USENIX Security Symposium.* USENIX Sec. 2022. ISBN: 978-1-939133-31-1.

[19] Bugzilla. *Improve initiator info to network requests.* `https://bugzilla.mozilla.org/show_bug.cgi?id=1230922`. 2022.

[20] BuiltWith. *Privacy Compliance Usage Distribution in the Top 1 Million Sites.* `https://web.archive.org/web/20201021075918/https://trends.builtwith.com/widgets/privacy-compliance/`. 2022.

[21] Sam Burnett and Nick Feamster. "Encore: Lightweight Measurement of Web Censorship with Cross-Origin Requests". In: *ACM Conference on Special Interest Group on Data Communication.* SIGCOMM. 2015. DOI: `10.1145/2785956.2787485`.

[22] Michael Butkiewicz, Harsha V Madhyastha, and Vyas Sekar. "Understanding website complexity: measurements, metrics, and implications". In: *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference.* 2011. DOI: `10.1145/2068816.2068846`.

[23] Katherine S Button, John PA Ioannidis, Claire Mokrysz, Brian A Nosek, Jonathan Flint, Emma SJ Robinson, and Marcus R Munafò. “Power failure: why small sample size undermines the reliability of neuroscience”. In: *Nature reviews neuroscience* 14.5 (2013). DOI: 10.1038/nrn3475.

[24] Stefano Calzavara, Tobias Urban, Dennis Tatang, Marius Steffens, and Ben Stock. “Reining in the Web’s Inconsistencies with Site Policy”. In: *Symposium on Network and Distributed System Security*. NDSS. 2021. DOI: 10.14722/ndss.2021.23091.

[25] Darion Cassel, Su-Chin Lin, Alessio Buraggina, William Wang, Andrew Zhang, Lujo Bauer, Hsu-Chun Hsiao, Limin Jia, and Timothy Libert. “OmniCrawl: Comprehensive Measurement of Web Tracking With Real Desktop and Mobile Browsers”. In: *Proceedings on Privacy Enhancing Technologies* 2.1 (2022). DOI: 10.2478/popets-2022-0012.

[26] Iain Chalmers, Michael B Bracken, Ben Djulbegovic, Silvio Garattini, Jonathan Grant, A Metin Gülmezoglu, David W Howells, John PA Ioannidis, and Sandy Oliver. “How to increase value and reduce waste when research priorities are set”. In: *The Lancet* 383.9912 (2014). DOI: 10.1016/S0140-6736(13)62229-1.

[27] Lakshmi Prasanna Chitra and Ravikanth Satapathy. “Performance comparison and evaluation of Node.js and traditional web server (IIS)”. In: *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*. 2017. DOI: 10.1109/ICAMMAET.2017.8186633.

[28] Cookiepedia by OneTrust. *Largest Database of Pre-Categorized Cookies*. https://cookiepedia.co.uk/. 2022.

[29] cURL Development Team. *cURL – Command Line Tool and Library for Transferring data with URLs*. https://web.archive.org/web/20230823093850/https://curl.se/. Online; Accessed: 2023-08-23. 2023.

[30] Adrian Dabrowski, Georg Merzdovnik, Johanna Ullrich, Gerald Sendera, and Edgar Weippl. “Measuring Cookies and Web Privacy in a Post-GDPR World”. In: *Conference on Passive and Active Measurement*. PAM. 2019. DOI: 10.1007/978-3-030-15986-3_17.

[31] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. “A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research”. In: *ACM Transactions on Information Systems* 2.39 (2021). DOI: 10.1145/3434185.

[32] Ha Dao and Kensuke Fukuda. “Alternative to Third-Party Cookies: Investigating Persistent PII Leakage-Based Web Tracking”. In: *International Conference on Emerging Networking EXperiments and Technologies*. CoNEXT. 2021. DOI: 10.1145/3485983.3494860.

[33] Martin Degeling, Christine Utz, Christoper Lentzsch, Henry Hosseini, Florian Schaub, and Thorsten Holz. “We Value Your Privacy ... Now Take Some Cookies: Measuring the GDPR’s Impact on Web Privacy”. In: *Symposium on Network and Distributed System Security*. NDSS. 2019. DOI: 10.1007/s00287-019-01201-1.

[34] Nurullah Demir, Matteo Große-Kampmann, Tobias Urban, Christian Wressnegger, Thorsten Holz, and Pohlmann Norbert. “Reproducibility and Replicability of Web Measurement Studies”. In: *International Conference on World Wide Web*. TheWebConf. 2022. DOI: `10.1145/3485447.3512214`.

[35] Nurullah Demir, Matteo Große-Kampmann, Tobias Urban, Christian Wressnegger, Thorsten Holz, and Norbert Pohlmann. “Reproducibility and Replicability of Web Measurement Studies”. In: *Proceedings of the ACM Web Conference 2022*. WWW ’22. Virtual Event, Lyon, France: Association for Computing Machinery, 2022. ISBN: 9781450390965. DOI: `10.1145/3485447.3512214`.

[36] Nurullah Demir, Jan Hörnermann, Matteo Große-Kampmann, Tobias Urban, Thorsten Holz, Norbert Pohlmann, and Christian Wressnegger. “On the Similarity of Web Measurements Under Different Experimental Setups”. In: *Proceedings of the 23nd ACM Internet Measurement Conference*. IMC ’23. Montreal, Canada: Association for Computing Machinery, 2023.

[37] Nurullah Demir, Daniel Theis, Tobias Urban, and Norbert Pohlmann. “Towards Understanding First-Party Cookie Tracking in the Field”. In: *Proc. of GI Sicherheit*. 2022.

[38] Nurullah Demir, Tobias Urban, Norbert Pohlmann, and Christian Wressnegger. “A Large-Scale Study of Cookie Banner Interaction Tools and Their Impact on Users’ Privacy”. In: *Proceedings on Privacy Enhancing Technologies*. PoPETS ’24 2024 (2024).

[39] Nurullah Demir, Tobias Urban, Kevin Wittek, and Norbert Pohlmann. “Our (in)Secure Web: Understanding Update Behavior of Websites and Its Impact on Security”. In: *Conference on Passive and Active Measurement*. PAM. 2021. ISBN: 978-3-030-72581-5. DOI: `10.1007/978-3-030-72582-2_5`.

[40] Clemens Deußer, Steffen Passmann, and Thorsten Strufe. “Browsing Unicity: On the Limits of Anonymizing Web Tracking Data”. In: *IEEE Symposium on Security and Privacy*. S&P. 2020. DOI: `10.1109/SP40000.2020.00018`.

[41] EasyList. *EasyList Cookie List*. `https://web.archive.org/web/20230822121511/https://secure.fanboy.co.nz/fanboy-cookiemonster.txt`. 2023.

[42] EasyList. *EasyPrivacy*. `https://easylist.to/easylist/easylist.txt`. 2021.

[43] EasyList. *EasyPrivacy*. `https://easylist.to/easylist/easylist.txt`. Used list as of 03/30/2022; version 202203300945. 2022.

[44] Steven Englehardt and Arvind Narayanan. “Online Tracking: A 1-Million-Site Measurement and Analysis”. In: *ACM Conference on Computer and Communications Security*. CCS. 2016. DOI: `10.1145/2976749.2978313`.

[45] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W. Felten. “Cookies That Give You Away: The Surveillance Implications of Web Tracking”. In: *International Conference on World Wide Web*. TheWebConf. 2015. DOI: `10.1145/2736277.2741679`.

[46] Jamie Ericson, Masoud Mohammadian, and Fabiana Santana. "Analysis of Performance Variability in Public Cloud Computing". In: *2017 IEEE International Conference on Information Reuse and Integration (IRI)*. 2017. DOI: 10.1109/IRI.2017.47.

[47] European Commission. *Data protection in the EU*. https://web.archive.org/web/20230806125752/https://commission.europa.eu/law/law-topic/data-protection/data-protection-eu_en. 2023.

[48] European Commission. *Ecodesign for Sustainable Products Regulation*. https://web.archive.org/web/20230820124937/https://commission.europa.eu/energy-climate-change-environment/standards-tools-and-labels/products-labelling-rules-and-requirements/sustainable-products/ecodesign-sustainable-products-regulation_en. Accessed: 20 August 2023. European Commission, 2023.

[49] Marjan Falahrastegar, Hamed Haddadi, Steve Uhlig, and Richard Mortier. "Tracking Personal Identifiers Across the Web". In: *Conference on Passive and Active Measurement*. PAM. 2016. DOI: 10.1007/978-3-319-30505-9_3.

[50] Qi Fan and Qingyang Wang. "Performance Comparison of Web Servers with Different Architectures: A Case Study Using High Concurrency Workload". In: *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*. 2015. DOI: 10.1109/HotWeb.2015.11.

[51] Tobias Flach, Nandita Dukkipati, Andreas Terzis, Barath Raghavan, Neal Cardwell, Yuchung Cheng, Ankur Jain, Shuai Hao, Ethan Katz-Bassett, and Ramesh Govindan. "Reducing Web Latency: The Virtue of Gentle Aggression". In: *SIGCOMM Comput. Commun. Rev.* 43.4 (Aug. 2013). ISSN: 0146-4833. DOI: 10.1145/2534169.2486014.

[52] Imane Fouad, Cristiana Santos, Feras Al Kassar, Nataliia Bielova, and Stefano Calzavara. "On Compliance of Cookie Purposes with the Purpose Specification Principle". In: *International Workshop on Privacy Engineering*. IWPE. 2020. DOI: 10.1109/EuroSPW51379.2020.00051.

[53] Hugh G. Gauch Jr. *Scientific Method in Brief*. Cambridge University Press, 2012. DOI: 10.1017/CBO9781139095082.

[54] Ghostery GmbH. *Ghostery—Privacy Ad Blocker*. https://www.ghostery.com/. 2023.

[55] GNU. *Wget – Command Line Tool*. https://web.archive.org/web/20230823093904/https://www.gnu.org/software/wget/. Online; Accessed: 2023-08-23. 2023.

[56] Peter Godfrey-Smith. *Theory and reality: An introduction to the philosophy of science*. University of Chicago Press, 2009. DOI: 10.1177/17456916211017098.

[57] Tom van Goethem, Ping Chen, Nick Nikiforakis, Lieven Desmet, and Wouter Joosen. "Large-Scale Security Analysis of the Web: Challenges and Findings". In: *International Conference on Trust and Trustworthy Computing*. TRUST. 2014. DOI: 10.1007/978-3-319-08593-7_8.

[58] Google. *Chrome Puppeteer*. https://web.archive.org/web/20230823093941/https://github.com/puppeteer/puppeteer. Online; Accessed: 2023-08-23. 2023.

[59] Google Inc. *BigQuery: Cloud Data Warehouse*. `https://cloud.google.com/bigquery`. 2021.

[60] Google Inc. *Chrome User Experience Report | Tools for Web Developers*. `https://developers.google.com/web/tools/chrome-user-experience-report`. Online; Accessed: 2020-06-08. 2020.

[61] Google, Inc. *Chromium*. `https://www.chromium.org/Home`. 2021.

[62] Google, Inc. *Fix lazy-loaded content*. `https://developers.google.com/search/docs/guides/lazy-loading?hl=en`. 2021.

[63] Colin M. Gray, Cristiana Santos, Nataliia Bielova, Michael Toth, and Damian Clifford. "Dark Patterns and the Legal Requirements of Consent Banners: An Interaction Criticism Perspective". In: *ACM SIGCHI Conference on Human Factors in Computing Systems*. CHI. 2021. DOI: `10.1145/3411764.3445779`.

[64] Florian Hantke, Stefano Calzavara, Moritz Wilhelm, Alvise Rabitti, and Ben Stock. "You Call This Archaeology? Evaluating Web Archives for Reproducible Web Security Measurements". In: *ACM Conference on Computer and Communications Security*. 2023.

[65] Raymond Hill. *uBlock Origin—Free, open-source ad content blocker*. `https://web.archive.org/web/20230823094255/https://ublockorigin.com/`. 2023.

[66] Maximilian Hils, Daniel W. Woods, and Rainer Böhme. "Measuring the Emergence of Consent Management on the Web". In: *ACM SIGCOMM Internet Measurement Conference*. IMC. 2020. DOI: `10.1145/3419394.3423647`.

[67] HTTP Archive. *About HTTP Archive*. `https://httparchive.org/about`. [Online; accessed 20. Oct. 2020]. 2020.

[68] HTTP Archive. *Methodology | The Web Almanac by HTTP Archive*. `https://almanac.httparchive.org/en/2020/methodology`. [Online; Accessed: 18. Jan. 2021]. 2020.

[69] HTTP Archive. *The HTTP Archive Tracks How the Web is Built*. `https://httparchive.org`. [Online; Accessed: 20. Oct. 2020]. 2020.

[70] HTTP Archive. *The HTTP Archive Tracks How the Web is Built*. `https://httparchive.org`. 2021.

[71] Xuehui Hu, Nishanth Sastry, and Mainack Mondal. "CCCC: Corralling Cookies into Categories with CookieMonster". In: *ACM Web Science Conference*. WebSci. 2021. DOI: `10.1145/3447535.3462509`.

[72] Xuehui Hu, Guillermo Suarez de Tangil, and Nishanth Sastry. "Multi-country Study of Third Party Trackers from Real Browser Histories". In: *IEEE European Symposium on Security and Privacy*. EuroS&P. 2020. DOI: `10.1109/EuroSP48549.2020.00013`.

[73] Cheng Huang, JiaYong Liu, Yong Fang, and Zheng Zuo. "A study on Web security incidents in China by analyzing vulnerability disclosure platforms". In: *Computers & Security* 58 (2016). DOI: `10.1016/j.cose.2015.11.006`.

[74] IAB Europe. *IAB Europe Transparency & Consent Framework Policies*. https://iabeurope.eu/iab-europe-transparency-consent-framework-policies/. 2022.

[75] Muhammad Ikram, Rahat Masood, Gareth Tyson, Mohamed Ali Kaafar, Noha Loizon, and Roya Ensafi. "The Chain of Implicit Trust: An Analysis of the Web Third-Party Resources Loading". In: *International Conference on World Wide Web*. WWW. International World Wide Web Conferences Steering Committee, 2019. DOI: 10.1145/3308558.3313521.

[76] Imperva, Inc. *Bad Bot Report 2020: Bad Bots Strike Back*. https://www.imperva.com/blog/bad-bot-report-2020-bad-bots-strike-back/. 2020.

[77] International Chamber of Commerce UK. *ICC UK Cookie guide*. https://www.cookielaw.org/wp-content/uploads/2019/12/icc_uk_cookiesguide_revnov.pdf. 2012.

[78] 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*. S&P. 2016. DOI: 10.1109/SP.2016.50.

[79] John Ioannidis. "Why Most Published Research Findings Are False". In: *PLOS Medicine* 2.8 (2005). DOI: 10.1371/journal.pmed.0020124.

[80] John PA Ioannidis. "How to make more published research true". In: *Revista Cubana de Información en Ciencias de la Salud (ACIMED)* 26.2 (2015). DOI: 10.1371/journal.pmed.1001747.

[81] John PA Ioannidis, Daniele Fanelli, Debbie Drake Dunne, and Steven N Goodman. "Meta-research: evaluation and improvement of research methods and practices". In: *PLoS biology* 13.10 (2015). DOI: 10.1371/journal.pbio.1002264.

[82] Costas Iordanou, Claudio Soriente, Michael Sirivianos, and Nikolaos Laoutaris. "Who is Fiddling with Prices? Building and Deploying a Watchdog Service for E-Commerce". In: *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. SIGCOMM '17. Los Angeles, CA, USA: Association for Computing Machinery, 2017. ISBN: 9781450346535. DOI: 10.1145/3098822.3098850.

[83] Umar Iqbal, Peter Snyder, Shitong Zhu, Benjamin Livshits, Zhiyun Qian, and Zubair Shafiq. "AdGraph: A Graph-Based Approach to Ad and Tracker Blocking". In: *IEEE Symposium on Security and Privacy*. S&P. 2020. DOI: 10.1109/SP40000.2020.00005.

[84] Nikhil Jha, Martino Trevisan, Luca Vassio, and Marco Mellia. "The Internet with Privacy Policies: Measuring The Web Upon Consent". In: *ACM Trans. Web* 16.3 (Sept. 2022). ISSN: 1559-1131. DOI: 10.1145/3555352.

[85] Leslie K John, George Loewenstein, and Drazen Prelec. "Measuring the prevalence of questionable research practices with incentives for truth telling". In: *Psychological science* 23.5 (2012). DOI: 10.1177/0956797611430953.

[86] Hugo Jonker, Benjamin Krumnow, and Gabry Vlot. "Fingerprint Surface-Based Detection of Web Bot Detectors". In: *European Symposium on Research in Computer Security*. ESORICS. 2019. DOI: 10.1007/978-3-030-29962-0_28.

[87] Jordan Jueckstock, Shaown Sarker, Peter Snyder, Aidan Beggs, Panagiotis Papadopoulos, Matteo Varvello, Ben Livshits, and Alexandros Kapravelos. "Towards Realistic and Reproducible Web Crawl Measurements". In: *International Conference on World Wide Web*. TheWebConf. 2021. DOI: 10.1145/3442381.3450050.

[88] Norbert L Kerr. "HARKing: Hypothesizing after the results are known". In: *Personality and social psychology review* 2.3 (1998). DOI: 10.1207/s15327957pspr0203_4.

[89] Daniel Kladnik. *I don't care about cookies 3.4.2—Get rid of cookie warnings from almost all websites!* https://www.i-dont-care-about-cookies.eu/. 2022.

[90] George Klees, Andrew Ruef, Benji Cooper, Shiyi Wei, and Michael Hicks. "Evaluating Fuzz Testing". In: *ACM Conference on Computer and Communications Security*. CCS. 2018. DOI: 10.1145/3243734.3243804.

[91] David Klein, Marius Musch, Thomas Barber, Moritz Kopmann, and Martin Johns. "Accept All Exploits: Exploring the Security Impact of Cookie Banners". In: *Proceedings of the 38th Annual Computer Security Applications Conference*. ACSAC '22. Austin, TX, USA: Association for Computing Machinery, 2022, pp. 911–922. ISBN: 9781450397599. DOI: 10.1145/3564625.3564647.

[92] Michael Kretschmer, Jan Pennekamp, and Klaus Wehrle. "Cookie Banners and Privacy Policies: Measuring the Impact of the GDPR on the Web". In: *ACM Transactions on the Web* 15.4 (2021). DOI: 10.1145/3466722.

[93] Chiara Krisam, Heike Dietmann, Melanie Volkamer, and Oksana Kulyk. "Dark Patterns in the Wild: Review of Cookie Disclaimer Designs on Top 500 German Websites". In: *European Symposium on Usable Security*. EuroUSEC. 2021. DOI: 10.1145/3481357.3481516.

[94] Benjamin Krumnow, Hugo Jonker, and Stefan Karsch. "How Gullible Are Web Measurement Tools? A Case Study Analysing and Strengthening OpenWPM's Reliability". In: *International Conference on Emerging Networking Experiments and Technologies*. CoNEXT. 2022. DOI: 10.1145/3555050.3569131.

[95] Raula Gaikovina Kula, Daniel M. German, Ali Ouni, Takashi Ishio, and Katsuro Inoue. "Do Developers Update their Library Dependencies?" In: *Empirical Software Engineering* 23.1 (2018). DOI: 10.1007/s10664-017-9521-5.

[96] Oksana Kulyk, Annika Hilt, Nina Gerber, and Melanie Volkamer. ""This Website Uses Cookies": Users' Perceptions and Reactions to the Cookie Disclaimer". In: *European Workshop on Usable Security*. EuroUSEC. 2018. DOI: 10.14722/eurousec.2018.23012.

[97] Deepak Kumar, Zane Ma, Zakir Durumeric, Ariana Mirian, J. Alex Mason Joshua a nd Halderman, and Michael Bailey. "Security Challenges in an Increasingly Tangled Web". In: *International Conference on World Wide Web*. WWW. 2017. DOI: 10.1145/3038912.3052686.

[98] Pierre Laperdrix, Nataliia Bielova, Benoit Baudry, and Gildas Avoine. "Browser Fingerprinting: A Survey". In: *ACM Transactions on the Web* 14.2 (2020). DOI: 10.1145/3386040.

[99] Tobias Lauinger, Abdelberi Chaabane, Sajjad Arshad, William Robertson, Christo Wilson, and Engin Kirda. "Thou Shalt Not Depend on Me: Analysing the Use of Outdated JavaScript Libraries on the Web". In: *Symposium on Network and Distributed System Security*. NDSS. 2017. DOI: `10.14722/ndss.2017.23414`.

[100] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. "Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation". In: *Symposium on Network and Distributed System Security*. NDSS. 2019. DOI: `10.14722/ndss.2019.23386`.

[101] Iain R. Learmonth, Gurshabad Grover, and Mallory Knodel. *Guidelines for Performing Safe Measurement on the Internet*. `https://web.archive.org/web/20230821180012/https://www.ietf.org/id/draft-irtf-pearg-safe-internet-measurement-08.html`. Informational. Network Working Group, 2023.

[102] Jeffrey T Leek and Leah R Jager. "Is most published research really false?" In: *Annual Review of Statistics and Its Application* 4 (2017). DOI: `10.1146/annurev-statistics-060116-054104`.

[103] Kirill Levchenko, Amogh Dhamdhere, Bradley Huffaker, Kc Claffy, Mark Allman, and Vern Paxson. "Packetlab: A Universal Measurement Endpoint Interface". In: *Proceedings of the 2017 Internet Measurement Conference*. IMC '17. London, United Kingdom: Association for Computing Machinery, 2017. ISBN: 9781450351188. DOI: `10.1145/3131365.3131396`.

[104] Bohan Li, Yongxiang Cai, Shuying Deng, and Zongyi He. "The Strategy of Personal Customization and Method of Collecting Professional Dynamic Information". In: *Journal of Physics: Conference Series*. JPCS. 2020. DOI: `10.1088/1742-6596/1626/1/012034`.

[105] Majestic. *The Majestic Million – The million domains we find with the most referring subnets*. `https://majestic.com/reports/majestic-million`. 2022.

[106] Matthew C Makel, Jonathan A Plucker, and Boyd Hegarty. "Replications in psychology research: How often do they really occur?" In: *Perspectives on Psychological Science* 7.6 (2012). DOI: `10.1177/1745691612460688`.

[107] Sourena Maroofi, Maciej Korczyński, and Andrzej Duda. "Are You Human? Resilience of Phishing Detection to Evasion Techniques Based on Human Verification". In: *ACM SIGCOMM Internet Measurement Conference*. IMC. 2020. DOI: `10.1145/3419394.3423632`.

[108] Célestin Matte, Nataliia Bielova, and Cristiana Santos. "Do Cookie Banners Respect my Choice? : Measuring Legal Compliance of Banners from IAB Europe's Transparency and Consent Framework". In: *2020 IEEE Symposium on Security and Privacy (SP)*. 2020, pp. 791–809. DOI: `10.1109/SP40000.2020.00076`.

[109] MDN Web Docs. *Window.localStorage*. `https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage`. 2022.

[110] Robert Merget, Juraj Somorovsky, Nimrod Aviram, Craig Young, Janis Fliegenschmidt, Jörg Schwenk, and Yuval Shavitt. “Scalable Scanning and Automatic Classification of TLS Padding Oracle Vulnerabilities”. In: *USENIX Security Symposium.* USENIX Sec. 2019.

[111] Mozilla Foundation. *OpenWPM on GitHub.* `https://github.com/mozilla/OpenWPM`. 2021.

[112] Marcus R Munafò, Brian A Nosek, Dorothy VM Bishop, Katherine S Button, Christopher D Chambers, Nathalie Percie du Sert, Uri Simonsohn, Eric-Jan Wagenmakers, Jennifer J Ware, and John Ioannidis. “A manifesto for reproducible science”. In: *Nature human behaviour* 1 (2017). DOI: `10.1038/s41562-016-0021`.

[113] Keshab Nath, Sourish Dhar, and Subhash Basishtha. “Web 1.0 to Web 3.0 - Evolution of the Web and its various challenges”. In: *2014 International Conference on Reliability Optimization and Information Technology (ICROIT).* 2014. DOI: `10.1109/ICROIT.2014.6798297`.

[114] National Academies of Sciences, Engineering, and Medicine. *Reproducibility and Replicability in Science.* Washington, DC: The National Academies Press, 2019. DOI: `10.17226/25303`.

[115] National Institute of Standards and Technology. *Official Common Platform Enumeration (CPE) Dictionary.* `https://nvd.nist.gov/products/cpe`. [Online; accessed 19. Oct. 2020]. 2020.

[116] Viet Hung Nguyen and Fabio Massacci. “The (Un)Reliability of NVD Vulnerable Versions Data: An Empirical Experiment on Google Chrome Vulnerabilities”. In: *ACM Asia Conference on Computer and Communications Security.* AsiaCCS. 2013. DOI: `10.1145/2484313.2484377`.

[117] Arian Akhavan Niaki, Shinyoung Cho, Zachary Weinberg, Nguyen Phong Hoang, Abbas Razaghpanah, Nicolas Christin, and Phillipa Gill. “ICLab: A Global, Longitudinal Internet Censorship Measurement Platform”. In: *IEEE Symposium on Security and Privacy.* S&P. 2020. DOI: `10.1109/SP40000.2020.00014`.

[118] Ninja Cookie. *Have you had enough of cookie banners? Forget about them! Ninja Cookie will take care of these and can say “no” to them for you!* `https://ninja-cookie.com/`. 2022.

[119] Midas Nouwens, Rolf Bagge, Janus Bager Kristensen, and Clemens Nylandsted Klokmose. “Consent-O-Matic: Automatically Answering Consent Pop-Ups Using Adversarial Interoperability”. In: *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems.* CHI. 2022. DOI: `10.1145/3491101.3519683`.

[120] Midas Nouwens, Ilaria Liccardi, Michael Veale, and Lalana Karger David a nd Kagal. “Dark Patterns after the GDPR: Scraping Consent Pop-Ups and Demonstrating Their Influence”. In: *ACM SIGCHI Conference on Human Factors in Computing Systems.* CHI. 2020. DOI: `10.1145/3313831.3376321`.

[121] Łukasz Olejnik, Claude Castelluccia, and Artur Janc. “Why Johnny Can’t Browse in Peace: On the Uniqueness of Web Browsing History Patterns”. In: *Proceedings on Privacy Enhancing Technologies*. PETS. 2012.

[122] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. “Website Fingerprinting at Internet Scale”. In: *Symposium on Network and Distributed System Security*. NDSS. 2016. DOI: 10.14722/ndss.2016.23477.

[123] Emmanouil Papadogiannakis, Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. “User Tracking in the Post-Cookie Era: How Websites Bypass GDPR Consent to Track Users”. In: *International Conference on World Wide Web*. WWW. 2021. DOI: 10.1145/3442381.3450056.

[124] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos Markatos. “Cookie Synchronization: Everything You Always Wanted to Know But Were Afraid to Ask”. In: *International Conference on World Wide Web*. WWW. 2019. DOI: 10.1145/3308558.3313542.

[125] Vern Paxson. “Strategies for Sound Internet Measurement”. In: *ACM SIGCOMM Internet Measurement Conference*. IMC. 2004. DOI: 10.1145/1028788.1028824.

[126] Playwright. *Fast and reliable end-to-end testing for modern web apps*. https://web.archive.org/web/20230806130807/https://playwright.dev/. 2023.

[127] Tom Preston-Werner. *Semantic Versioning 2.0.0*. https://semver.org/. Online; Accessed: 20. Oct. 2020. 2020.

[128] James H Price and Judy Murnan. “Research Limitations and the Necessity of Reporting them”. In: *American Journal of Health Education* 35.2 (2004). DOI: 10.1080/19325037.2004.10603611.

[129] Proton Technologies AG. *ProtonVPN: Secure and Free VPN service for protecting your privacy*. https://protonvpn.com/. 2021.

[130] Elissa M. Redmiles, Sean Kross, and Michelle L. Mazurek. “How I Learned to Be Secure: A Census-Representative Survey of Security Advice Sources and Behavior”. In: *ACM Conference on Computer and Communications Security*. CCS. 2016. DOI: 10.1145/2976749.2978307.

[131] *Retraction Watch - Tracking retractions as a window into the scientific process*. https://web.archive.org/web/20230802055744/https://retractionwatch.com/. Accessed on: 02 August 2023. 2023.

[132] Richard Roberts, Yaelle Goldschlag, Rachel Walter, Taejoong Chung, Alan Mislove, and Dave Levin. “You are who you Appear to be: A Longitudinal Study of Domain Impersonation in TLS Certificates”. In: *acm-ccs*. CCS. 2019. DOI: 10.1145/3319535.3363188.

[133] Paula T Ross and Nikki L Bibler Zaidi. “Limited by our Limitations”. In: *Perspectives on Medical Education* 8.4 (2019). DOI: 10.1007/s40037-019-00530-x.

[134] Christian Rossow, Christian J. Dietrich, Grier Grier, Christian Kreibich, Vern Paxson, Norbert Pohlmann, Herbert Bos, and Maarten van Steen. “Prudent Practices for Designing Malware Experiments: Status Quo and Outlook”. In: *IEEE Symposium on Security and Privacy.* 2012. DOI: `10.1109/SP.2012.14`.

[135] Sebastian Roth, Stefano Calzavara, Moritz Wilhelm, Alvise Rabitti, and Ben Stock. “The Security Lottery: Measuring Client-Side Web Security Inconsistencies”. In: *USENIX Security Symposium.* USENIX Sec. 2022.

[136] Shauvik Roy Choudhary, Husayn Versee, and Alessandro Orso. “WEBDIFF: Automated identification of cross-browser issues in web applications”. In: *2010 IEEE International Conference on Software Maintenance.* 2010. DOI: `10.1109/ICSM.2010.5609723`.

[137] Morteza Safaei Pour, Christelle Nader, Kurt Friday, and Elias Bou-Harb. “A Comprehensive Survey of Recent Internet Measurement Techniques for Cyber Security”. In: *Computers & Security* 128 (2023). ISSN: 0167-4048. DOI: `10.1016/j.cose.2023.103123`.

[138] Iskander Sanchez-Rola, Matteo Dell’Amico, Platon Kotzias, Davide Balzarotti, Leyla Bilge, Pierre-Antoine Vervier, and Igor Santos. “Can I Opt Out Yet? GDPR and the Global Illusion of Cookie Control”. In: *ACM Asia Conference on Computer and Communications Security.* AsiaCCS. 2019. DOI: `10.1145/3321705.3329806`.

[139] Will Keeling. *selenium-wire 4.3.1.* `https://pypi.org/project/selenium-wire/`. 2021.

[140] Saptak Sengupta, Tom Van Goethem, and Nurullah Demir. *The 2021 Web Almanac: HTTP Archive’s annual state of the web report.* The Web Almanac. ISBN: 9798985709742. 2021.

[141] M. Zubair Shafiq, Amir R. Khakpour, and Alex X. Liu. “Characterizing caching workload of a large commercial Content Delivery Network”. In: *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications.* 2016. DOI: `10.1109/INFOCOM.2016.7524379`.

[142] Muhammad Shahzad, Muhammad Zubair Shafiq, and Alex X. Liu. “A Large Scale Exploratory Analysis of Software Vulnerability Life Cycles”. In: *International Conference on Software Engineering.* ICSE. 2012. DOI: `10.5555/2337223.2337314`.

[143] shopware AG. *Updating Shopware.* `https://docs.shopware.com/en/shopware-5-en/update-guides/updating-shopware`. [Online; accessed 20. Oct. 2020]. 2020.

[144] Patrick E. Shrout and Joseph L. Rodgers. “Psychology, Science, and Knowledge Construction: Broadening Perspectives from the Replication Crisis”. In: *Annual Review of Psychology* 69.1 (2018). DOI: `10.1146/annurev-psych-122216-011845`.

[145] Joseph P. Simmons, Leif D. Nelson, and Uri Simonsohn. “False-Positive Psychology: Undisclosed Flexibility in Data Collection and Analysis Allows Presenting Anything as Significant”. In: *Psychological Science* 22.11 (2011). PMID: 22006061. DOI: `10.1177/0956797611417632`.

[146] Alexander Sjösten, Peter Snyder, Antonio Pastor, Panagiotis Papadopoulos, and Benjamin Livshits. “Filter List Generation for Underserved Regions”. In: *International Conference on World Wide Web*. TheWebConf. 2020. DOI: 10.1145/3366423.3380239.

[147] Software Freedom Conservancy. *SeleniumHQ Browser Automation*. https://web.archive.org/web/20230823094129/https://www.selenium.dev/. Online; Accessed: 2023-08-23. 2023.

[148] Léna Soler, Emiliano Trizio, Thomas Nickles, and William C. Wimsatt. *Characterizing the Robustness of Science: After the Practice Turn in Philosophy of Science*. Springer Science & Business Media, 2012. DOI: 10.1007/978-94-007-2759-5.

[149] International Organization for Standardization. *ISO/IEC 27002:2022*. https://web.archive.org/web/20230820085428/https://www.iso.org/standard/75652.html. Accessed on: 20 August 2023. 2022.

[150] State of California Department of Justice. *California Consumer Privacy Act (CCPA)*. https://web.archive.org/web/20230806125953/https://oag.ca.gov/privacy/ccpa. 2023.

[151] Jonathan AC Sterne and George Davey Smith. “Sifting the evidence—what’s wrong with significance tests?” In: *Physical therapy* 81.8 (2001). DOI: 10.1136/bmj.322.7280.226.

[152] Ben Stock, Martin Johns, Marius Steffens, and Michael Backes. “How the Web Tangled Itself: Uncovering the History of Client-Side Web (In)Security”. In: *USENIX Security Symposium*. SEC. 2017. DOI: 10.5555/3241189.3241265.

[153] Victoria Stodden, Peixuan Guo, and Zhaokun Ma. “Toward reproducible computational research: an empirical analysis of data and code policy adoption by journals”. In: *PloS one* 8.6 (2013). DOI: 10.1371/journal.pone.0067111.

[154] Super Agent. *No more Pop-Ups! Privacy can be Simple*. https://www.super-agent.com/. 2022.

[155] Samaneh Tajalizadehkhoob, Tom Van Goethem, Maciej Korczyński, Arman Noroozian, Rainer Böhme, Tyler Moore, Wouter Joosen, and Michel van Eeten. “Herding Vulnerable Cats: A Statistical Approach to Disentangle Joint Responsibility for Web Security in Shared Hosting”. In: *ACM Conference on Computer and Communications Security*. CCS. 2017. DOI: 10.1145/3133956.3133971.

[156] The Electronic Frontier Foundation. *Privacy Badger*. https://privacybadger.org/. 2023.

[157] The World Bank. *Individuals using the Internet (% of population)*. https://web.archive.org/web/20230806124016/https://data.worldbank.org/indicator/IT.NET.USER.ZS. 2023.

[158] Marco Torchiano, Filippo Ricca, and Alessandro Marchetto. “Are Web Applications More Defect-prone than Desktop Applications?” In: *International Journal on software tools for technology transfer* 13.2 (2011). DOI: 10.1007/s10009-010-0182-6.

[159] Michael Toth, Nataliia Bielova, and Vincent Roca. "On Dark Patterns and Manipulation of Website Publishers by CMPs". In: *Proceedings on Privacy Enhancing Technologies*. PoPETs. 2022. DOI: 10.56553/popets-2022-0082.

[160] Martino Trevisan, Danilo Giordano, Idilio Drago, and Ali Safari Khatouni. "Measuring HTTP/3: Adoption and Performance". In: *Mediterranean Communication and Computer Networking Conference*. MedComNet. 2021. DOI: 10.1109/MedComNet52149.2021.9501274.

[161] Tobias Urban, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. "Beyond the Front Page: Measuring Third Party Dynamics in the Field". In: *International Conference on World Wide Web*. TheWebConf. 2020. DOI: 10.1145/3366423.3380203.

[162] Tobias Urban, Dennis Tatang, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. "Measuring the Impact of the GDPR on Data Sharing". In: *ACM Asia Conference on Computer and Communications Security*. AsiaCCS. 2020. DOI: 10.1145/3320269.3372194.

[163] Christine Utz, Martin Degeling, Sascha Fahl, Florian Schaub, and Thorsten Holz. "(Un)informed Consent: Studying GDPR Consent Notices in the Field". In: *ACM Conference on Computer and Communications Security*. CCS. 2019. DOI: 10.1145/3319535.3354212.

[164] Phani Vadrevu and Roberto Perdisci. "What You See is NOT What You Get: Discovering and Tracking Social Engineering Attack Campaigns". In: *Proceedings of the Internet Measurement Conference*. IMC '19. Amsterdam, Netherlands: Association for Computing Machinery, 2019. ISBN: 9781450369480. DOI: 10.1145/3355369.3355600.

[165] Steven Van Acker, Daniel Hausknecht, and Andrei Sabelfeld. "Measuring Login Webpage Security". In: *Symposium on Applied Computing*. SAC. 2020, pp. 1753–1760. DOI: 10.1145/3019612.3019798.

[166] Tom Van Goethem and Nurullah Demir. *The 2022 Web Almanac: HTTP Archive's annual state of the web report*. The Web Almanac. ISBN: 9798985709766. 2022.

[167] Tom Van Goethem, Nurullah Demir, and Barry Pollard. *The 2020 Web Almanac: HTTP Archive's annual state of the web report: Security*. The Web Almanac. ISBN: 9798985709728. 2020.

[168] Kami Vaniea and Yasmeen Rashidi. "Tales of Software Updates: The Process of Updating Software". In: *ACM SIGCHI Conference on Human Factors in Computing Systems*. CHI. 2016. DOI: 10.1145/2858036.2858303.

[169] Antoine Vastel, Walter Rudametkin, Romain Rouvoy, and Xavier Blanc. "FP-Crawlers: Studying the Resilience of Browser Fingerprinting to Block Crawlers". In: *Workshop on Measurements, Attacks, and Defenses for the Web*. MADWeb. 2020. DOI: https://dx.doi.org/10.14722/madweb.2020.23010.

[170] David Y. Wang, Stefan Savage, and Geoffrey M. Voelker. "Cloak and Dagger: Dynamics of Web Search Cloaking". In: *ACM Conference on Computer and Communications Security*. CCS. 2011. DOI: 10.1145/2046707.2046763.

[171] Wappalyzer. *Identify technology on websites—Wappalyzer.* https://www.wappalyzer.com. Online; Accessed: 2020-06-08. 2020.

[172] Web Almanac By HTTP Archive. *Privacy: Compliance with privacy regulations.* https://almanac.httparchive.org/en/2022/privacy#consent-management-platforms. 2022.

[173] WebPageTest. *WPO-Foundation/webpagetest: Official repository for WebPageTest.* https://web.archive.org/web/20230806130817/https://github.com/WPO-Foundation/webpagetest. 2023.

[174] Michael Weisberg. "Robustness Analysis". In: *Philosophy of Science* 73.5 (2006), pp. 730–742. DOI: 10.1086/518628.

[175] William J. Welch. "Construction of Permutation Tests". In: *Journal of the American Statistical Association* 85.411 (1990). DOI: 10.1080/01621459.1990.10474929.

[176] Daniel W. Woods and Rainer Böhme. "The Commodification of Consent". In: *Computers and Security* 115.C (2022). DOI: 10.1016/j.cose.2022.102605.

[177] WordPress. *Configuring Automatic Background Updates.* https://wordpress.org/support/article/configuring-automatic-background-updates. [Online; accessed 20. Oct. 2020]. 2019.

[178] Peter Wurzinger, Christian Platzer, Christian Ludl, Engin Kirda, and Christopher Kruegel. "SWAP: Mitigating XSS Attacks Using a Reverse Proxy". In: *ICSE Workshop on Software Engineering for Secure Systems.* IWSESS. 2009. DOI: 10.1109/IWSESS.2009.5068456.

[179] Zhiju Yang and Chuan Yue. "A Comparative Measurement Study of Web Tracking on Mobile and Desktop Environments". In: *Proceedings on Privacy Enhancing Technologies.* PETS. 2020. DOI: 10.2478/popets-2020-0016.

[180] 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: *International Conference on World Wide Web.* TheWebConf. 2020. DOI: 10.1145/3366423.3380104.

[181] Yixin Zou, Abraham H. Mhaidli, Austin McCall, and Florian Schaub. ""I've Got Nothing to Lose": Consumers' Risk Perceptions and Protective Actions after the Equifax Data Breach". In: *Symposium on Usable Privacy and Security.* SOUPS. 2018. DOI: 10.5555/3291228.3291245.

Appendix A

Supplementary Information

A.1. Availability of Data & Code Artifacts

To foster future research, we release our code, measurement data, evaluation, and other supplementary information for all studies presented in the thesis openly online:

Reproducibility and Replicability Study (Section 3.1)

`https://doi.org/10.5445/IR/1000142435`

Similarity Study (Section 3.2)

`https://doi.org/10.35097/1719`

Web Security Study (Chapter 4)

`https://doi.org/10.5445/IR/1000142476`

Web Privacy Study (Chapter 5)

`https://doi.org/10.35097/1708`

A.2. Appendix for Chapter 3

In this Appendix we provide additional information (e.g., experimental setup, additional results and discussions) for Chapter 3.

A.2.1. Experimental Setup

To measure the impact of different experimental setups, we built a pipeline that enables us to compare measurement results based on different setups. To allow comparability, parallel

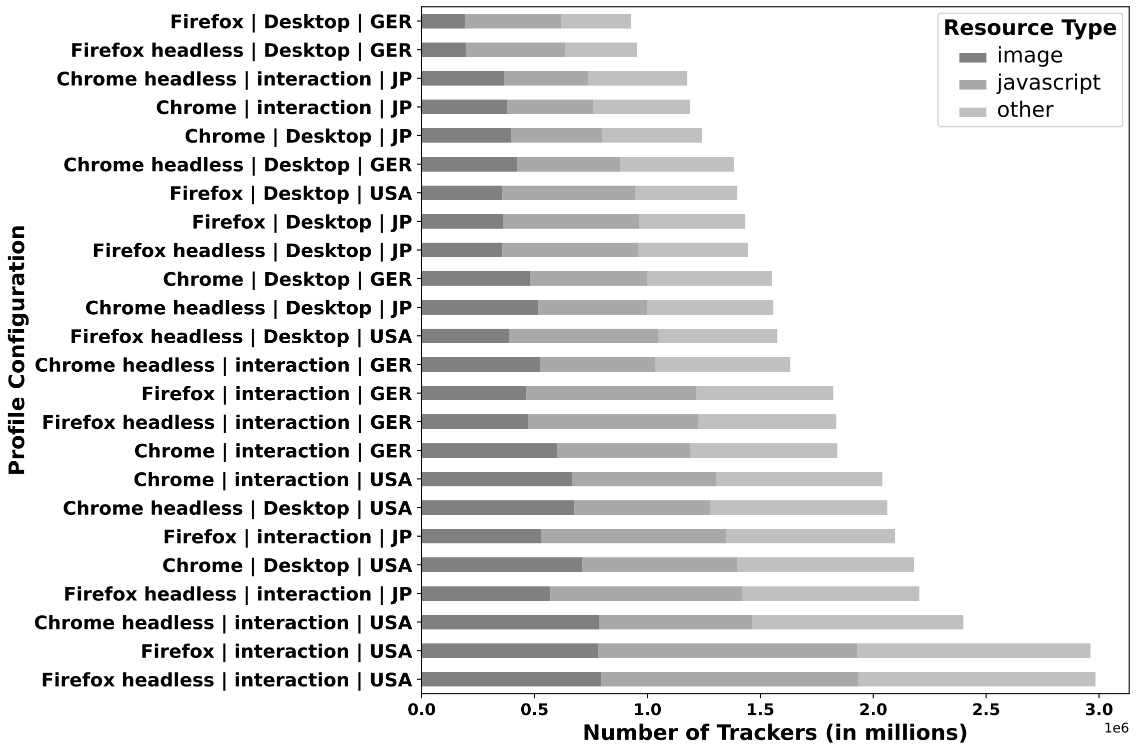

Figure A.1.: Number of identified trackers for each profile and their resource type.

page visits across all defined profiles are essential. Our measurement setup consists of different virtual machines (VMs) orchestrated by a "commander" to organize parallel page visits. For all VMs, we use *Ubuntu 20.04* as operating system and do not pass the GPU to them, which can impact fingerprinting scripts [1]. Each of the worker VMs conducts the measurement according to a specific profile (see C5–C10 below).

As an initial step, the commander assembles the set of URLs that should be visited during the experiment based on the heuristic described in C1 and C2 (see below). For our experiment, this had happened on 06/24/2021, three days before we started the measurement. At the beginning of the measurement, the commander starts one VM for each of the 24 browser profiles. The VMs will query the commander for batches of sites ($n = 10$) to analyze. All VMs receive an identical list of sites and pages, such that all VMS conduct measurements in the same order, starting with the site's landing page. Once all managed VMs are ready to start their measurements, the commander issues a signal to start the experiment as *stateless* coordinated crawls of the provided URLs. Each VM starts 10 browsers (one for each site) in parallel using the defined profile. Once the analysis of a page is finished, the same browser instance is moving to the next page of the site. Once the results have been stored, the VM will query for the next batch of URLs and wait until the commander tells all VMs to start the analysis. To conduct this large-scale measurement and to host the virtual machines with the necessary resources, we use a server which is equipped with 256GB RAM, a *AMD 7542, 2.90GHz* CPU with 32 cores, and a 10 Gbps network interface.

We supplied each VM with 10GB RAM, five CPUs, and 40GB of hard disc space to cache the results before sending them to the commander.

A.2.1.1. Dataset

C1 In our analysis, we use the Tranco list generated on 06/23/2021, which is available at `https://tranco-list.eu/list/ZGPG/15000` [100]. We use the top 10k sites to build our website dataset.

C2 Tranco lists only contain sites (eTLD+1). Therefore, we used the following heuristic to identify the landing page by defining a protocol to use. We used four prefixes (`http[s]://[www.]`, starting with the https variants) and test if the resulting URL is reachable. If so, we added the first identified site to our corpus. Once we determined the seed URLs for our crawl, we visited each of them and randomly chose up to 25 first-party links (recursively if necessary), which we used for our measurement run. Selecting subsites is essential since they often show a different behavior compared to the landing pages [161, 8]. To avoid incorporating duplicated URLs, we always ignore each identified link's anchor part and omit links that could result in redirects (e.g., links including `http://` in the path).

C3 We make the list of pages and sites openly available (see App. A.1).

C4 For our continuous measurement, we visit the top 1k sites from our website corpus (18,377 distinct pages) daily (starting at midnight) throughout our experiment (from 07/08/2021 to 07/19/2021).

A.2.1.2. Experimental Design

C5 We use four different browser types (Chrome, Chrome headless, Firefox, Firefox headless) in our study. We use the popular *OpenWPM* Framework [44] (v0.15.0 – Firefox version 88) to perform the Firefox-based measurements and Chromium [61] to perform the Chrome-based measurements, respectively.

C6 Regarding adjustments to *OpenWPM*, we built a wrapper that feeds the pages to visit and which extracts the measurement results. Hence, the wrapper does not affect the functionality of the framework. Regarding changes to *OpenWPM*, we use two custom commands provided by the framework (1) for logging visits and (2) to simulate "user interaction" (see C10). Regarding flexible options of the framework, we used the `native` display mode, which enables the GUI browser and disable it for the the `headless` mode. Otherwise, we used the standard configuration of the platform. For our Chrome setup, we aim to use setups similar to other studies [4]. Hence, we utilize *Selenium* [147] to build our Chrome-based crawler (version: 91). When implementing the crawler and page visiting strategy, we oriented at the parameters used in *OpenWPM* to allow more realistic comparison (e.g., each page visit is done in a new tab or waiting for resources to be loaded). To conduct the headless crawl, we pass the *headless* argument to Selenium.

C7 Aside the named adjustments, we did not extend the browsers.

C8 To disguise our crawler, we modified the standard *Selenium* parameters based on the findings of Jonker et al. , who empirically studied which techniques are used in practice to detect such crawlers [86] (e.g., changing the user agent). Otherwise, we implemented simulated user interaction (see C10).

C9 We make the used framework publicly available (see App. A.1).

C10 We use two approaches to simulate user interaction: (1) no interaction ("none") and (2) mimicking artificial user interaction ("user interaction"). Thus, for (1) we do not interact with the website at all. In profile (2), once the browser loads the page, we wait for 30 seconds or until the page finished loading and then simulate three *page down* keystrokes followed by three *Tab* keystrokes, and finally an *end* keystroke with minimal periods of delay in between.

C11 When visiting a page, we wait until a page has finished loading, close the browser, and move on to the next page. We use a timeout of 30 seconds for each page visit (standard configuration of *OpenWPM*), after which the visit will be terminated (e.g., to avoid slow websites).

C12 We choose three geolocations for our measurement: (1) Germany (EU), (2) Japan (AS), and (3) the United States (NA). We choose these continents since the majority of the analyzed papers focuses on them. To simulate the geolocations, we used *ProtonVPN* [129].

C13 We altered browsers resolution to `1366x768` and the user agents to `Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/ 91.0.4472.77 Safari/537.36` for *Chrome* and `Mozilla/5.0 (X11; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0` for *Firefox*.

C14 For *OpenWPM*, after crawling a site, our wrapper (described in C6) extracts the HTTP traffic from the database provided by *OpenWPM*. The Chrome-based crawlers collect the HTTP traffic with the help of *selenium-wire* [139]. The results of the measurements are pushed into a *Google BigQuery* database [59].

A.2.1.3. Evaluation

C15 Our (raw) results are publicly available (see App. A.1).

C16 We elaborate on our findings in Sec. 3.1.2.2.

C17 & C18 We discuss limitations and ethics in Sec. 3.1.3.

A.2.2. Experimental Design and Additional Results for Similarity Study

We develop our measurement framework along with recent findings on how to run robust Web measurements [34, 87]. Our measurement setup is based on the openly available framework of Demir et al. that implements a best-effort approach to conduct parallel Web measurements [34]. In a nutshell, the framework consists of a commander machine administering the experiments and several clients (i.e., virtual machines) that run a distinct browser profile. The commander orchestrates the entire measurement process by supplying the URLs to visit to all clients at once. To achieve parallelism, the commander waits until each client visited all pages before providing a new set of pages. Hence, not all page visits are parallel, but the site visits are started simultaneously. Note that the deviation in the visits in our measurement is acceptable (avg: 46 seconds, SD: 111 seconds). The large standard deviation is caused by pages that timeout (e.g., by a slowly loading ad) in one profile but not in another. While the measured deviation could impact the content of a webpage (e.g., a blog publishes a post with in that time), we assume that our approach is a suitable compromise between scalability and parallelism. This design choice reduces the crawling time as, for example, timeouts might be smoothed. It is worth noting that each virtual machine must run the technology used to visit the pages (i.e., the tools to analyze the page visits). Thus, the framework provides the flexibility to compare different measurement setups. Out-of-the-box, the framework consolidates the results from each VM and stores them in a *BigQuery* [59] database.

A.2.2.1. General Measurement Configuration

To increase the repeatability and reproducibility of our result, we elaborate in the following on the general configuration options we used in our experiments. We conducted all measurements from the same public IP address associated with an university network, which might introduce some bias into our results. Furthermore, each VM runs 15 browser instances in parallel. We configured a timeout for each page visit of 30 seconds (similar to previous works [34, 161, 162]). Other works used a longer time out (e.g., [87, 45]) but the effects of different timeouts have not been studied in detail yet and could be addressed in future work. To allow fair scalability of our experiment, in which we visit over 1.7M pages, we choose to resort to a shorter timeout. We do not perform any consequent experiments to understand a timely impact on the results because previous work already highlighted such effects [34], and we assume this will propagate to our results. Web measurements can be performed *stateless*, which means that the state of the browser is reset after each page visit (e.g., cookies set) or *stateful* (the state of the browser is preserved between page visits). Both options come with different up- and downsides. We choose to use a *stateless* approach, which means that the order of the site visits does not impact the results. Furthermore, this means that our study will provide a lower-bound of the problem.

A.2.2.2. Visualization of the Comparison Approach

Figure A.2 provides an overview of our comparison approach. The red boxes illustrate the recursive and the blue boxes illustrate the vertical approach.

Vertical: (1) across all trees, we compare the entire loading dependencies of the node e (dashed blue squares), which is not present in tree #3; (2) we cross-compare the direct partners of each node in a branch (dotted blue squares). *Horizontal*: we analyze the nodes on depth one (dashed red square) and recursively the children of nodes that appear in

Table A.1.: Observed size of the trees and similarities of children and parents across the buckets.

#	Bucket	mean nodes	child sim	parent sim
1	1-5k	448	.71	.72
2	5,001–10k	434	.71	.71
3	10,001–50k	427	.71	.72
4	50,001–250k	417	.69	.66
5	250,001–500k	369	.70	.67

more than one tree (dotted red squares—the figure shows only one example to increase readability).

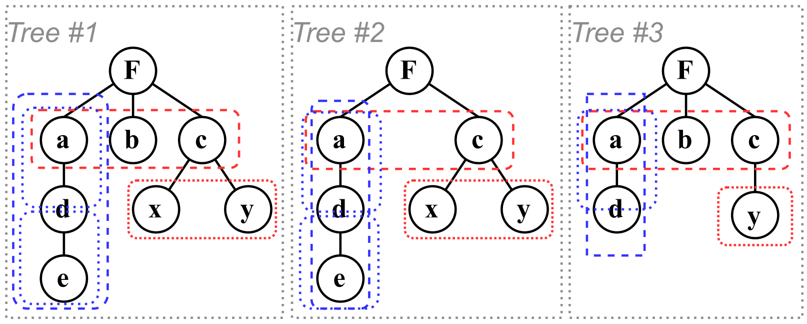

Figure A.2.: Depiction of our vertical (blue) and horizontal (red) comparison approach.

A.2.2.3. Additional results

In the following we provide additional results regarding our study Section 3.2.

Understanding Implications of Site Popularity. Previous work has shown that a website's popularity impacts different phenomena [18, 161]. In the following, we analyze if and how the popularity of websites, in terms of their rank on the Tranco list, affects the changes in a tree. This analysis aims to understand if popular sites act differently regarding the complexity and similarity of the observed trees. Table A.1 provides an overview of the sizes of the trees in the different buckets and shows the similarities in the parent and child nodes across each bucket. Given the plain numbers, the trees of popular sites have more nodes, but the similarity values are nearly identical. These figures show that popular sites produce larger trees, but there is no difference in the resulting similarities of the trees. However, the ANOVA test found statistical significance between the total number of nodes and the sites' rank and between the rank and the observed children and parent similarities (p-value < 0.001). Nevertheless, the effect size is marginal ($\eta^2 = .0004$), meaning there is a statistical significance effect, but it is practically negligible. Thus, a site's rank does not impact the similarity when measuring it multiple times.

Similarity of Different Resource Types. In Section 3.2.2.2 we show that the resource type can significantly affect the similarity of the observed trees. Figure A.3 provides an

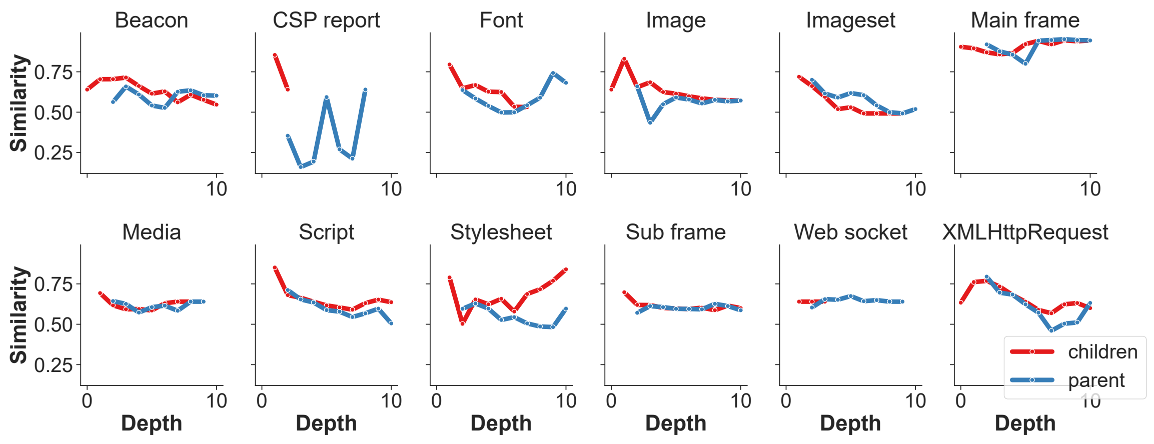

Figure A.3.: Average similarity of children and parent nodes for different resource types based on their tree depth.

overview of the average similarity of children and parent nodes of all resource types based on their depth in the trees. One can observe how the similarity of nodes' children and parents differs based on the content type and the depth. Overall, we record that the similarity for specific content types stays stable (e.g., web socket); at the same time, we see that the similarity for some content types (e.g., script) changes drastically based on the observed depth.

A.3. Appendix for Chapter 4

In this Appendix, we provide additional results for Chapter 4.

A.3.1. Overview of the Top Identified CWEs

In this appendix, we show our findings related to identified CWEs. Table A.2 lists the most common CWEs on websites that we identified in the last measurement run (June 2020). While the vulnerability *Cross-site Scripting* (XSS) occurs in almost all websites, a closer analysis of the same measurement point (M#18) shows that only 28% of software is vulnerable to this vulnerability.

A.3.2. Average Age of the Top 20 Software Used in Website Ranking

Figure A.4 shows the most popular software types, their average age (in month), and the rank of the websites on which they are used. We record that most of the widely used software on the web is often very old. We also found that the average age of utilized software on a website is unrelated to its popularity, according to the *Tranco* list [100].

Table A.2.: Top 10 vulnerabilities in our last measurement point (M#18) by relative frequency on websites.

Vulnerability Type (CWE)	Relative Frequency
CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	0.92
CWE-20 Improper Input Validation	0.32
CWE-400 Uncontrolled Resource Consumption	0.27
CWE-200 Exposure of Sensitive Information to an Unauthorized Actor	0.24
CWE-476 NULL Pointer Dereference	0.24
CWE-601 URL Redirection to Untrusted Site ('Open Redirect')	0.22
CWE-125 Out-of-bounds Read	0.22
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer	0.20
CWE-787 Out-of-bounds Write	0.19
CWE-190 Integer Overflow or Wraparound	0.17
CWE-284 Improper Access Control	0.17

Table A.3.: Some examples of vulnerabilities identified on analyzed websites that run outdated software.

Software	CVE	CVE Publication	CWE	CVSS	Public exploit	Vuln. Websites	Total usage
jQuery	CVE-2020-11023	04.2020	XSS	4.3	✗	3.98M	4M
Apache	CVE-2017-7679	06.2017	Buffer Over-read	7.5	✓	0.26M	0.46M
PHP	CVE-2015-8880	05.2016	Double free	10	✓	0.45M	0.46M
PHP	CVE-2016-2554	03.2016	Buffer Over-read	10	✓	0.23M	0.46M
WordPress	CVE-2018-20148	12.2018	Deserialization of Untrusted Data	7.3	✓	0.18M	0.46M
WordPress	CVE-2019-20041	12.2019	Improper Input Validation	7.3	✗	0.31M	0.46M

A.3.3. Case Studies

Table A.3 illustrates the most common CVE entries identified in our study. *CVE-2020-11023* is the most common vulnerability with the severity "MEDIUM" – based on our last measurement. Some of the vulnerabilities require certain functions or enabled functions (e.g., CVE-2017-7679 for *Apache* requires *mod_mime* and CVE-2016-2554 for *PHP* requires file uploading functionality) In some cases, the running software requires interaction between more than one component to abuse an exploit. The listed vulnerabilities for *WordPress* and vulnerability CVE-2015-8880 for *PHP* do not require any interaction or enabled features and can be exploited directly.

A.4. Appendix for Chapter 5

In this Appendix, we provide additional results for Chapter 5.

A.4.1. Success Rates of the analyzed Extension in the Manual Test

Table A.4 provides an overview of the respective success rates of the analyzed extensions. We assume that the 'interaction' was successful if a prompted cookie banner disappears automatically (or is not present) when we visit a page with an installed extension. Otherwise,

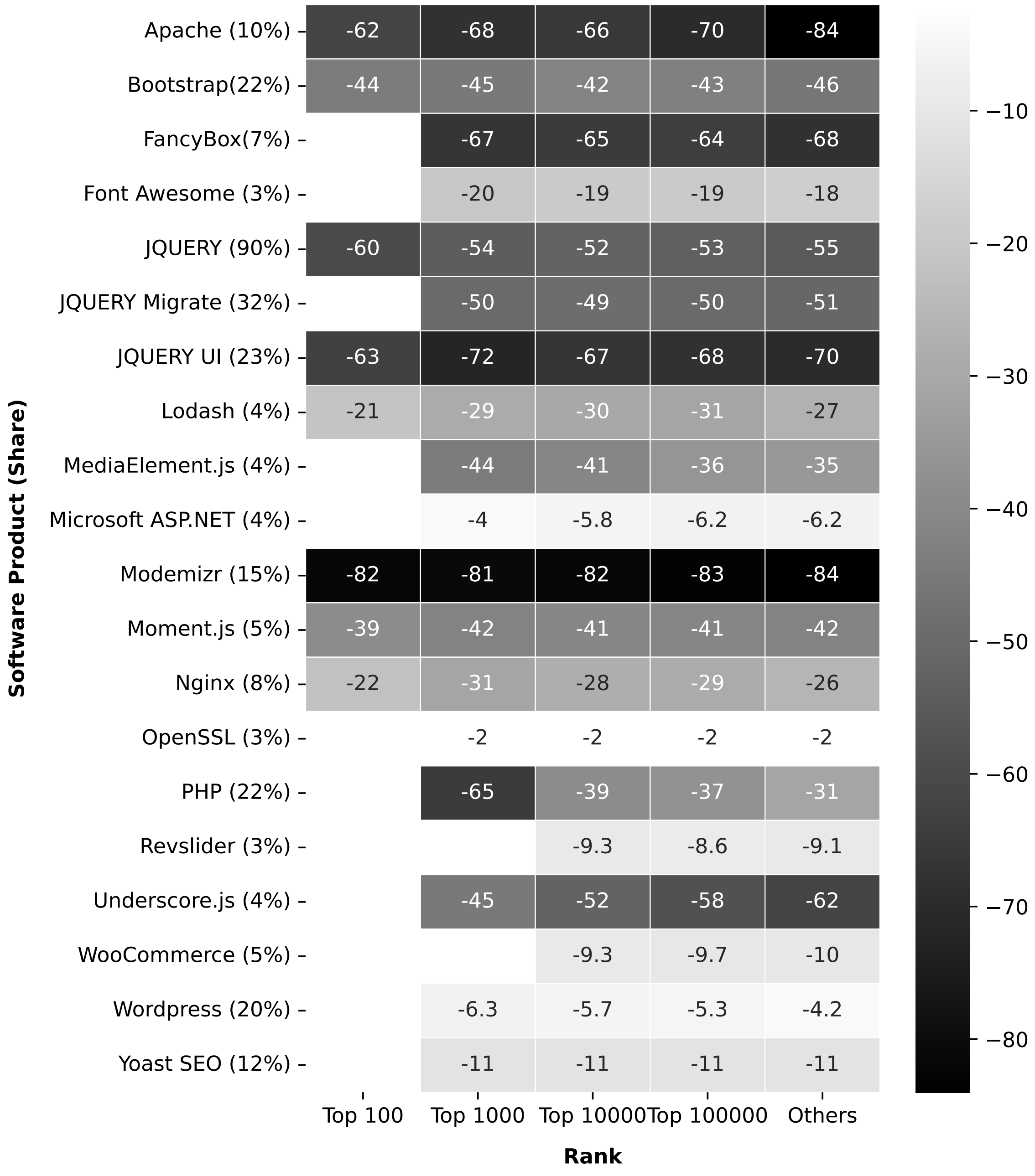

Figure A.4.: Average age (in month) of the top 20 used software by website ranking. The share of software in our dataset is shown in brackets – Blank cells: no website identified in the corresponding ranking.

we assume that the extension does not work on the visited page. Note that the extension *CookieBlock* does not interact with cookie banners but actively deletes cookies of specific categories. Overall, we see that the analyzed extension have different success rates in interaction with the banner, but no extension broke a page in our manual experiment.

Table A.4.: Overview of the success rates of the extension to interact with the 19 cookie banners in our manual test.

Name	Successful interaction		Unsuccessful interaction	
I don't care about cookies	18	95%	1	5%
Consent-O-Matic	10	53%	9	47%
Ninja Cookie	12	63%	2	37%
SuperAgent	9	47%	10	53%
CookieBlock	—	—	—	—

A.4.2. Failure Rates of the Crawler

We only include sites in our experiment if at least eight profiles successfully crawled them to ensure a fair and meaningful comparison of all tools. This filtering resulted in the exclusion of 36% of all sites. It is worth noting that this (high) rate is solely attributed to the combination of the profiles—each profile has a failure rate of <15%. More precisely, profile #9 has the highest (15%) and profile #6 the lowest (13%) failure rate; the mean failure rate is 14%. These numbers are typical for large-scale Web measurements [34, 24].

A.4.3. Cookies in the Different Profiles

Figure A.5 shows the number of cookies present in each profile. It is evident from the figure that each profile has a distinct distribution of cookies, with profile #5 having the lowest number of cookies and profile #9 having the highest. These results highlight that the different browser extensions can have a considerable effect on the type and quantity of cookies set by a website.

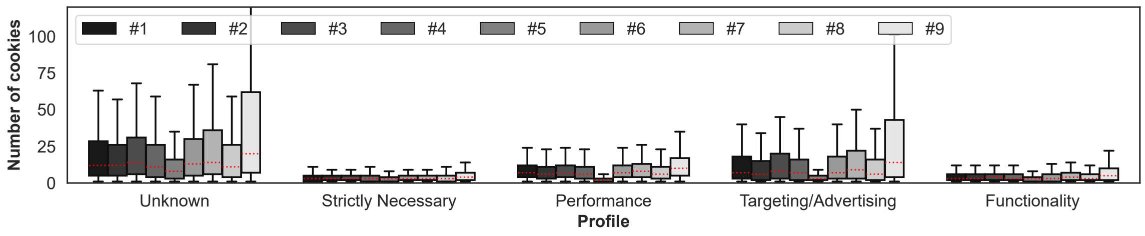

Figure A.5.: Number of cookies in the different profiles by cookie usage category. To increase readability, the y-axis is cut at 100. The upper whisker of the category "Unknown" in profile #9 is around 150.

A.4.4. Configurations of *Consent-O-Matic*

The *Consent-O-Matic* extension offers, in addition to the standard configuration, six configuration options to allow or deny different types of cookies: (1) Preferences and Functionality;

(2) Performance and Analytics; (3) Information Storage and Access; (4) Content selection, delivery, and reporting; (5) Ad selection, delivery, and reporting; and (6) Other purposes. Thus, we analyzed seven different configurations of the extension. One should note that the default configuration of *Consent-O-Matic* aims to reject all cookies that are not necessary for the page to work. To experiment with a reasonable time frame, we analyzed each option individually and not a combination of them, which would result in analyzing 63 configurations ($\binom{6}{6} + \binom{6}{5} + \binom{6}{4} + \binom{6}{3} + \binom{6}{2} + \binom{6}{1} = 63$). Furthermore, we choose the top 7,000 sites from the utilized Tranco list and randomly sampled 15 pages (cf. Section 5.2.1), visited them with our measurement framework and filtered the pages to analyze according to our filtering rules (cf. Section 5.2). We conducted the experiments between 05/08/2023 and 05/12/2023 from a European IP address (Germany) using *Amazon Web Services AWS*.

A.4.5. Statistical Effects of Different Consent-O-Matic Configurations

Table A.5 shows the p-values when computing the statistical significance, using the Kruskal-Wallis test, of the different *Consent-O-Matic* profiles regarding the presence of tracking requests and different cookie types. The lighter-gray fields highlight the profile combinations for which we found statistical significance. Especially the profile that limits the use of "Performance" cookies (`Perfm.`) shows a notably different behavior than most other profiles. For the configurations that allow functional cookies (`Func.`) and "Information Storage and Access" cookies (`Infor.`), we get mixed results meaning that they show statistical difference to roughly half of the configurations. Most profiles (i.e., the baseline, the default configuration, reports, ads, and other) show a very similar (i.e., almost no statistical significance) behavior compared to the other configurations. These comparisons concerning the statistically significant impact on the metrics of interest show that most configuration options have little impact on them.

Table A.5.: Kruskal-Wallis test results for tracking request numbers and cookie types across different *Consent-O-Matic* configurations.

	Baseline	**Default**	**Func.**	**Perfm.**	**Infor.**	**Report**	**Ads**	**Other**
Baseline	—	1.00	0.06	0.00	1.00	1.00	1.00	1.00
Default	1.00	—	0.02	0.00	1.00	1.00	1.00	1.00
Func.	0.06	0.02	—	1.00	0.04	0.46	0.00	0.01
Perfm.	0.00	0.00	1.00	—	0.00	0.05	0.00	0.00
Infor.	1.00	1.00	0.04	0.00	—	1.00	1.00	1.00
Report	1.00	1.00	0.46	0.05	1.00	—	1.00	1.00
Ads	1.00	1.00	0.00	0.00	1.00	1.00	—	1.00
Other	1.00	1.00	0.01	0.00	1.00	1.00	1.00	—

A.4.6. Locations of the Repeated Measurements

Table A.6 shows the three locations that we used for our repeated control experiments. We performed all experiments successive using different *Amazon Web Services* instances during April 2023.

Table A.6.: Locations and times of the repeated measurements.

#	City	Country	Start Date	End Date
1	Frankfurt	DEU	04/21/23	04/24/23
2	Paris	FRA	04/24/23	04/28/23
3	Stockholm	SWE	04/28/23	05/01/23

A.4.7. Analysis on Cookies that Could Not be Classified

In the following, we analyze the cookies that could not be identified by *Cookeipedia* [28], and test their possible impacts on our analysis. Overall, we could not classify 43% of the cookies, which means that *Cookeipedia* could not identify their purpose (cf. Section 5.2.5). To better understand if a manual classification of some of the observed unclassified cookies is feasible and would notably enhance our analysis, we analyze their distribution, overall occurrence, and further properties. We first test the characteristics of these cookies and find that 65% of the unclassified cookies are third-party cookies and 86% of them are session cookies. A deeper analysis of these cookies shows that almost a third (37%) of the unclassified cookies have unique names. Table A.7 provides an overview of the most common unclassified cookies. Overall, a manual classification of the top 5 cookies would increase the number of *classified cookies by 2,7%*, and a manual classification of the top 10 cookies would increase the number by 4%, and a classification of the top 100 cookies would increase the number by 15%. It is worth noting that 2 of the top 5 cookies are probably "functional" cookies, meaning all extensions would accept them. Therefore, we did not conduct any manual classification on these cookies as it would have a minimal effect on the overall results.

Table A.7.: Overview of the top unclassified cookies, their overall occurrence in our dataset, an example value, and their expected functionality based on an Internet search.

Name	Occurrence	Ex. value	Functionality
`__cf_bm`	1.08%	fwhRuDEX7J...	Functional cookie to detect bots.
`c`	0.59%	1662508350	Timestamp of unknown propose
`CMTS`	0.36%	1150	Unknown
`A3`	0.35%	d=AQABBIsK...	Unknown
`li_gc`	0.32%	MTswOzE2Nj...	Functional cookie to store consent preferences.