On the Similarity of Web Measurements Under Different Experimental Setups

Nurullah Demir
Institute for Internet Security
KASTEL Security Research Labs
Karlsruhe Institute of Technology
demir@internet-sicherheit.de

Jan Hörnemann
AWARE7 GmbH
jan@aware7.de

Matteo Große-Kampmann
Rhine-Waal University of Applied Sciences
AWARE7 GmbH
matteo@aware7.de

Tobias Urban
Institute for Internet Security
secunet Security Networks AG
urban@internet-sicherheit.de

Norbert Pohlmann
Institute for Internet Security
pohlamnn@internet-sicherheit.de

Thorsten Holz
CISPA Helmholtz Center for Information Security
holz@cispa.de

Christian Wressnegger
KASTEL Security Research Labs
Karlsruhe Institute of Technology
c.wressnegger@kit.edu

ABSTRACT

Measurement studies are essential for research and industry alike better understand the Web's inner workings and help quantify specific phenomena. Performing such studies is demanding due to the dynamic nature and size of the Web. Designing and setting up an experiment is a complex task, and many factors might affect the results. However, while several works have independently observed differences in the outcome of an experiment (e.g., the number of observed trackers) based on the measurement setup, it is unclear what causes such deviations. This work investigates the reasons for these differences by visiting 1.7M webpages with five different measurement setups. Based on this investigation, we build 'dependency trees' for each page and cross-compare the nodes in the trees. The results show that the measured trees differ considerably, that the cause of differences can be attributed to specific nodes, and that even identical measurement setups can produce different results.

CCS CONCEPTS

• **General and reference** → **Measurement**; *Empirical studies*; *Design*; • **Security and privacy**;

KEYWORDS

Web Measurements, Reproducibility, Robustness, Metascience

ACM Reference Format:
Nurullah Demir, Jan Hörnemann, Matteo Große-Kampmann, Tobias Urban, Norbert Pohlmann, Thorsten Holz, and Christian Wressnegger. 2023. On the Similarity of Web Measurements Under Different Experimental Setups. In *Proceedings of the 2023 ACM Internet Measurement Conference (IMC '23), October 24–26, 2023, Montreal, QC, Canada.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3618257.3624795

IMC '23, October 24–26, 2023, Montreal, QC, Canada

ACM ISBN 979-8-4007-0382-9/23/10…$15.00
https://doi.org/10.1145/3618257.3624795

1 INTRODUCTION

Modern websites are intricate and complex software applications that 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 [22, 25]. 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 [34] or HTTP headers [31]. Consequently, the same webpage could present different objects during Web measurement studies.

Researchers often make use of Web measurements to comprehend various phenomena, like Web tracking, security mechanisms, or the behavior of social media sites, that affect millions of users [1, 12, 15, 17, 23, 29, 35]. However, the dynamic nature of the Web poses a significant challenge to these measurements. Tools such as *OpenWPM* [19] 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 [2, 11, 14, 24, 31].

While previous studies have mainly explored the *effects* of different measurement setups, there is still no adequate explanation for *why* the results differ. This study bridges that gap by investigating the impact 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 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.

2 TERMINOLOGY AND BACKGROUND

In the following, we further elaborate on Web measurements in general (cf. Section 2.1) and discuss prior work on representing websites as trees (cf. Section 2.2). For this study, we use the term *site* to refer to the registerable part of a domain—often called the "extended Top Level Domain plus one" (eTLD+1). The term *page* refers to a unique URL or the document (e.g., HTML or JavaScript) located at that URL.

2.1 Web Measurement Experiments

Web measurements are indispensable for understanding the Web, its adherence to legislation, and the functioning of its vast ecosystems [16, 34, 35]. They illuminate Web content trends, the extent of vulnerabilities in technologies such as TLS [4, 28, 30], and infrastructural aspects such as SSL or HTTP/3 adoption [6, 33]. However, the complexity and dynamics of the Web make these measurements challenging. The dynamic content display mechanisms and variations in technology stacks complicate Web measurements [11]. Prior work has shown that slight setup changes can significantly influence the results, emphasizing the importance of understanding how various factors affect the outcomes [14]. The 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 7). This study examines the structure and dependencies of dynamic content.

2.2 Representing Websites as a Tree

With their complexity and dynamic content, websites require a uniform representation to conduct systematic analysis. One efficient representation is a tree that models all resources and their dependencies [22, 32, 34]. We build trees following previous work to enhance our work's comparability. The edges in a tree symbolize HTTP communication, and a node represents loaded content. 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 our work (cf. Section 3.2).

3 METHOD

In this section, we describe our experimental setup (cf. Section 3.1) and our approach to measuring the differences between webpages (cf. Section 3.2).

3.1 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 results we compare. Semi-parallel means that each profile runs on a separate VM and that the visits are synchronized on the site level, but differences might occur on the page level. For example, visits to `foo.com` start on all VMs simultaneously, but each VM visits all site pages independently. We develop our measurement framework in line with recent findings on how to run robust Web measurements [2, 27, 34, 37, 38]. We made this design choice to make our work comparable to previous works because currently the comparability of different works needs to be improved in our community [14]. Our setup is based on the openly available framework of Demir et al. that implements a best-effort approach to conduct parallel Web measurements [14]. We describe the framework used in Appendix C to foster reproducibility.

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. These tests were selected based on the characteristics inherent in our data. Specifically, our datasets exhibit non-normal distributions, necessitating non-parametric methods. Additionally, the independence of the groups being compared, whether paired or unpaired, further guided our choice of these particular tests. The chosen methods align with our data's underlying properties and structure, ensuring robust conclusions. We use a significance level of $\alpha = .05$ for all tests.

3.1.1 Experimental Design. In the following, we describe the configuration of the five different browser profiles we use in our experiment. We employ these profiles since these types have been used by previous works [14] to highlight existing comparability issues in our community. Using methods or profiles that others have not

used before could adversely affect 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 [19], to capture the traffic we are interested in. We choose to utilize *OpenWPM* because it is widely used in the Web measurement community and, therefore, serves as a good foundation to conduct our experiments. As the framework already has a rich feature set, we only made adjustments to mimic user interaction, as described later in this section. The named design choices regarding the measurement framework help increase our work's comparability and reproducibility.

Table 1 lists the profiles we use in our work. We cross-compare these five profiles to understand the differences in the analyzed pages. We chose the profiles based on different configurations commonly used in previous and related works [14]. 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. 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 [14], which could heavily impact comparability, and 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 [14, 24]. Additionally, we used the most recent stable Firefox browser 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. Still, our approach allows us to understand differences introduced by other browser versions. Furthermore, a ten-month-old browser should still be supported by most websites. We make our browser profiles and the crawling technology available at https://github.com/internet-sicherheit/On-the-Similarity-of-Web-Measurements-Under-Different-Experimental-Setups/ (cf. Appendix A).

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, aside 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 selected these keys as they will probably not load a different page. The timeout and the specific keystrokes have been used by prior work (e.g., [15]). Such mimicked user interaction also interferes with bot detection mechanisms so 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.

Table 1: Overview of the used profiles. Profile #2 and #3 use the same setup.

#	Name	Version	User 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

3.1.2 Website Dataset. The basis of the set of websites to analyze is the widely used quasi-standard Tranco list [26]. 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 [3, 14, 34]. We repeated the process recursively if the landing page did not hold enough links. We list all analyzed pages and sites in Appendix A.

3.2 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 website differences 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 (i.e., child node). This approach is similar to methods used by previous works [22, 25, 34], 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 that 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 a 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 slightly differs 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 6.

To build the trees, we make use of (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 the stack does not directly indicate the dependencies of requests but those 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 [8]. Thus, we handle them the same way as JavaScript dependencies. This artifact results from 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 [17, 19, 23, 29, 35]. Thus, analyzing such privacy-invasive requests allows us to put our results into perspective with other works. To identify the requests, we profit from the tracking filter list *EasyList* [18]). 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) and discuss limitations in Section 6.

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 dropping roughly 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 at least 89%, comparable to related approaches [10, 14]. Thus, the seemingly high dropping rate is due to the varying success of the crawlers in visiting the pages of interest successfully.

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 tree analysis'). 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 identify reoccurring objects

Table 2: High-level overview of the measured trees.

Tree	avg.	SD	min	max
nodes	84	99	1	12k
depth	3.6	2.2	0	30
breadth	44	58	1	12k

Node(s)...	
each present in X profiles (avg)	3.6
present in all profiles	52%
present in one profile	24%

in multiple trees with at least one child, we perform a similarity measurement of the children of these nodes. We repeat this step until we do not find any additional elements in at least two profiles. To assess the *vertical* similarity, we perform a bottom-up approach, 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 each child's entire dependency chain to understand the determinism 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 loads specific content. This approach provides a context-specific perspective on which nodes are loaded and by whom. Appendix D provides an overview of our 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. The "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 apply the *Jaccard index*, which 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 equal 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 similarity (sim.) categories [14]: *high* (sim. ≥ 0.8), *medium* ($0.3 <= \text{sim.} < 0.8$), and *low* (sim. < 0.3) We choose not to compute similarities of entire trees (e.g., using the *Hamming distance* [37]) but instead compute the similarity of the nodes present in the trees as it provides deeper insights into the changes in the relationships between the nodes. We analyze branches in multiple trees using set operations and the Jaccard index.

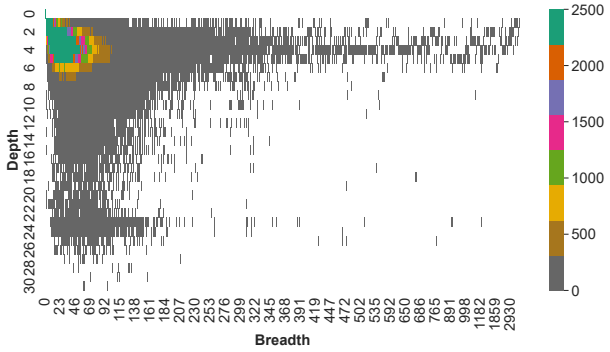

Figure 1: Distribution of the observed trees' depth/breadth. We cap the scaling at 2,500 to improve the readability.

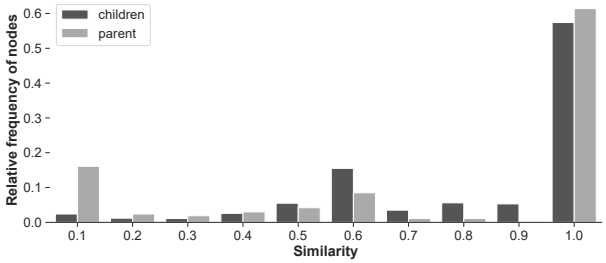

Figure 2: Distribution of the similarities of all observed nodes per tree and the similarities of parents.

4 RESULTS

Before detailing the results (Sections 4.1 to 4.4), 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). 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 (standard deviation (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. This step 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%).

General Structure, Size, and Differences of the Trees. The core of our analysis are the previously described dependency trees (cf. Section 3.2). In our experiment, we cross-compare five trees for the same page; 1.66M trees in total. Table 2 provides an overview of the measured trees, and Fig. 1 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 2), 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 are present in only one profile. The results show that when comparing two different profiles, 48% of the underlying data varies. On a high level, this indicates the severe impact of the Web's dynamic and the used crawler on measurement studies that we aim to understand better in this work.

4.1 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 for 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. Fig. 2 shows the 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). 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 expected 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 at the same depth. Based on this analysis, we determine that if a node appears in all trees, it will appear at the same depth, and that nodes 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 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). On a high level, these results indicate that third-party nodes do not occur as

Table 3: 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

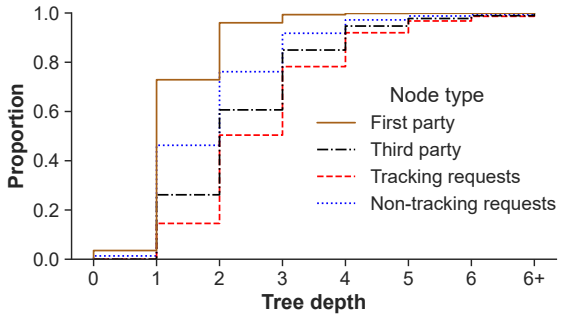

Figure 3: Volume of different types of nodes in the trees. Nodes after depth six are combined into one.

stable as first-party nodes do. These figures indicate that studies focusing on third-party elements (e.g., 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, Fig. 3 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.

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.

4.2 Node Relations Across Different Trees

We have shown that differences in the observed trees exist, which raises the question of how they are introduced.

Cross-comparing Parents in Tree Branches. This part discusses the similarities of dependency chains to provide an understanding of the differences in the loading dependencies in the trees. We use the term *dependency chains* to describe all (grand)parents of each node. More technically, a dependency 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 dependency 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 the nodes (on average 12% per profile) have a unique dependency 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 dependency chains. Thus, on a high level, the results show that overlaps between the trees exist but that considerable parts vary.

The presented results show that loading dependencies are only sometimes stable. In the following, we shed light on *why* and *where* the dependency 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 dependency 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.

As dependency chains tend to be non-deterministic, analyzing which kind of nodes—in terms of resource types—introduce variations is interesting. In the following, we test which resource types are always loaded by the same dependency chain at a level deeper than one. Table 4a provides an overview of the most common resource types that are always loaded by the same dependency chains. In contrast, Table 4b shows the resource types with the lowest similarity, meaning they are often loaded by different 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 same dependency chain loads the vast majority of first-party notes (86%); in contrast, this only applies to 56% of third-party nodes. These results replicate for 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

Table 4: Effects of resource type on loading dependencies.

(a) Same dependency chain.

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
images	.25
Web sockets	.27
style sheets	.31
Web beacons	.34

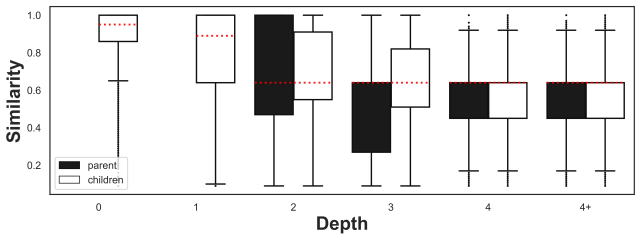

Figure 4: The similarity of children and parents. We combined all nodes occurring at a depth deeper than four ("4+") to increase readability.

users. For measurement studies, these findings implicate that high-level results, e.g., the presence of a specific node, can be compared. However, the reasons why these results occurred (e.g., why a node is present) can differ. Hence, studies regarding the ecosystem of a phenomenon can yield different results based on the setup used.

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 for assessing the expected 'measurement fluctuations' and indicate an experiment's accuracy. 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, when analyzing 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, these requests were triggered by different parent nodes in different trees (i.e., the dependency chains differ). The mean similarity of these nodes' parents is .33, which means that several resources 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. So far, we have 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. The average number of children on all depths larger than zero ranges between 0.2 and 1. This distinct drop is expected since many components cannot dynamically load additional objects. For example, an HTML image tag cannot load additional content besides 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). Appendix E details 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 (cf. Section 4.3 for further details). However, outlier nodes with several children, responsible for the tree's growth, exist on all levels. Thus, considerable parts of a tree are related to only a few nodes, which means that if they change, the tree itself changes substantially.

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 in the similarities, starting to increase again in deep branches. This observation is rooted in the fact that on deeper levels, only a few nodes exist ($n < 100$), which are loaded by a small set of nodes. Fig. 4 shows the decreasing similarity based on the depth, and Fig. 7 in Appendix G provides an overview of the similarity of children and parents for different resource types per depth. One result of this observation is that if a phenomenon of interest appears at deeper levels, it is possible that other measurements will not observe it and might draw different conclusions. 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.

(a) Resource types by average *parent* similarity.

(b) Resource types by average *child* similarity.

Figure 5: Distribution of the most common resource types based on the average similarity of all nodes on a page.

Understanding Implications of Resource Types. To gain an understanding of the effects of a node's type on the dissimilarities of a page, we examine the impact of its resource type on the similarity of its children and parents. Fig. 5a 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. On webpages with low parent similarity, the node types `image`, `script`, and `subframe` (e.g., iframes) make the highest relative share, which indicates that they are mainly responsible for dissimilarities. In contrast, Figure 5b provides an overview of children's similarities. Again, images show the highest similarity for child nodes. The Kruskal-Wallis test found a statistical significance that the resource type of a 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, and their variance introduction in the results tampers with a Web measurement experiment's comparability, reproducibility, or replicability.

> This section shows that the dependencies between the nodes 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 and how the entity type could affect a study.

4.3 First- and Third-Party Context

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

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 depths one and two, and they dominate third-party nodes (only) on these levels (depth 0: 99%; depth 1: 55%). To understand how consistently a first-party node appears on a page, 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 extent. Note that some changes, in our case, user interaction, can have far-reaching impacts on the results and therefore decrease the comparability of the results. These numbers validate that our framework, measurement approach, and analysis method are valid since they produce the expected results.

Implications on Third-party Level. While first-party nodes and their children are quite stable in their embedding, third-party nodes probably show a different behavior due to their high dynamics [34]. 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 the observed third-party nodes appear in all profiles and how their appearance frequency changes. Third-party nodes appear less stable across the different profiles than first-party nodes: Our analysis shows that the third-party nodes at depth one appear on average in 3.9 profiles. 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 node (i.e., at depth one) is similar across all profiles, while the subsequently loaded third-party nodes 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 third-party nodes have many 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 an 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.

This section showed distinct differences between nodes loaded in the first- and third-party context. The similarity of nodes in the first-party context is high, while, 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 4.4 and 5.3). 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.

4.4 Assessing Setup Implications

We have shown that the analyzed sites introduce notable variance in the performed measurements. While this is challenging itself, it is essential to understand which impact the measurement setup has on the outcome of an experiment. Table 5 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. Table 1). 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 ($\leq$5), the trees are highly similar (mean: .92), but that the similarity decreases on the deeper levels (mean: .75).

Table 5: 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	4,562
3	Sim2	19.34M	13.19M	3.20M	29	4,258
4	NoAction	14.53M	9.25M	1.91M	30	4,953
5	Headless	19.39M	13.22M	3.20M	30	4,562

Table 6: Profile differences compared to profile #2 (`Sim1`). ∗: Starting at depth two. ✚: 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) ∗	.71	.70	.70	.71
child similarity (mean) ✚	.83	.84	.74	.84

Table 6 provides an overview of the observed differences between all profiles. We compare the results with profile #2 (`Sim1`) because it serves as a reference to a profile often used in related works [14]. The results show a distinct difference in the relative number of perfectly similar nodes (i.e., nodes that appear at the same depth in both profiles). It is important to note that even if the numbers across all profiles are similar (except for profile `NoAction`), the impact on the results is different. It is also not the case that the same node across all measurements shows 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 the comparability of two measurements 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 6). Thus, using an outdated browser version has similar effects on the comparability of an experiment as 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* mimic user interactions (profile #4; `NoAction`). Similar to previous works [14, 34], we find that simulating user interactions causes 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 *fewer* 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, profile `NoAction` shows

the highest variation (i.e., fewer perfectly similar nodes and more nodes with no similarity) across metrics except first-party nodes (cf. Table 6). This observation shows that introducing simple simulated user interactions can stabilize a measurement.

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 6) 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 significant impact of using the headless mode [14], in contrast to (older) works that found such impact [1].

The results presented in this section indicate that the outcome of a measurement can differ significantly depending on the utilized browser configuration. Even when 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.

5 CASE STUDIES

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

5.1 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.

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 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.

5.2 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 [5]. We analyzed cookies because they are widely studied (e.g., [10, 12, 20, 35]). 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), while 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 finding is surprising because these are hard-coded attributes that would not be expected to differ.

5.3 Tracking Requests

Finally, we analyze the implications of the results for tracking requests, an 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. The similarity of parents of tracking nodes (.53; SD: .27; min: .09; max: 1;) is lower than the similarity of parents of non-tracking nodes, and 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 that they mostly appear in the upper parts. 9% of the observed tracking nodes 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. Looking at the parents of the tracking nodes, we find that tracking requests are often triggered by other trackers (65%), which are primarily loaded in a third-party context (82%). On average, we see that 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 tracker shows that 46% of these nodes are triggered by JavaScript resources, 34% by subframes, and 15% directly by mainframes.

6 LIMITATIONS

Our method has limitations that we discuss in the following. Aside from artificially mimicked user interaction, we do not interact with the visited pages. Our approach does not consider content that is only displayed based on a user action, in a particular use case (e.g., triggering of a payment service) or in a specific state of the user's browser (e.g., being logged in at a service). In addition, our crawler does not interact with content notices. However, since all measurements are performed from the same location, these limitations affect all profiles similarly, and comparing page visits is still suitable. In this sense, our results are not complete but can rather be seen as a lower bound. 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. Another limitation of our approach is that different URLs might look the same after purging the parameters, even if they load additional 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 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 considerably change our findings and takeaways. Adding blocklists could also result in a more distorted measurement because lists like *EasyPrivacy* do not focus on tracker blocking [13].

7 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 [2]. Furthermore, Demir et al. analyze the state of the art of how the Web security and privacy community performs and documents their experiments [14]. 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 [24]. 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 [11]. Yang and Yue develop *WTPatrol* and, in a comparative measurement study of Web tracking on 23,310 websites with mobile and desktop version webpages, find that mobile Web tracking has unique characteristics [37]. Vastel et al. find that 291 websites of the Alexa top 10k block crawlers effectively use fingerprinting [36]. Most recently, Calzavara et al. show that archive-based measurements might provide a solution to the reproducibility problem, and they develop best practices for future measurements [9]. Finally, several other works discussed that small changes in a measurement setup could significantly affect the outcome (e.g., visiting subsites) [2, 3, 27, 34, 37, 38]. 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.

8 CONCLUSION AND TAKEAWAYS

In this work, we performed a large-scale Web measurement study to understand the effects of different experimental setups commonly used. 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 that must be considered when conducting and comparing Web measurement studies. Furthermore, the results show that differences in the 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. The main takeaways from our study are:

(1) Future work should investigate how to assess "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 whether the phenomenon of interest is present in the dynamic (e.g., ads) or static (e.g., HTTP headers) content of a page is vital for planning the experiments.
(4) Our approach confirms that researchers should use different profiles and execute multiple measurements to assess the potential of 'randomized' findings.

ACKNOWLEDGMENTS

The authors gratefully acknowledge funding from the *Helmholtz Association* (HGF) within topic "46.23 Engineering Secure Systems," the *Federal Ministry Economic Affairs and Climate Action of Germany* (grants 01MK20008E "Service-Meister," 16KIS1629, 16KIS1628K, and 16KIS1900 "UbiTrans," and 16KIS1648 "DigiFit"), and the the *Federal Office for Information Security of Germany* (grants 01MO23033B "5Guide" and 01MO23025B "Pentest-5GSec").

REFERENCES

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

[2] Syed Suleman Ahmad, Muhammad Daniyal Dar, Muhammad Fareed Zaffar, Narseo Vallina-Rodriguez, and Rishab Nithyanand. 2020. Apophanies or Epiphanies? How Crawlers Impact Our Understanding of the Web. In *International Conference on World Wide Web (WWW)*. https://doi.org/10.1145/3366423.3380113

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

[4] 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. 2016. DROWN: Breaking TLS Using SSLv2. In *USENIX Security Symposium (USENIX Sec.)*.

[5] Adam Barth. 2011. *HTTP State Management Mechanism*. RFC 6265. Internet Engineering Task Force. 1–37 pages. https://tools.ietf.org/html/rfc6265

[6] Adrian Bermudez-Villalva, Mirco Musolesi, and Gianluca Stringhini. 2020. 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)*. 494–499. https://doi.org/10.1109/EuroSPW51379.2020.00072

[7] Dino Bollinger, Karel Kubicek, Carlos Cotrini, and David Basin. 2022. Automating Cookie Consent and GDPR Violation Detection. In *USENIX Security Symposium (USENIX Sec.)*.

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

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

[10] Stefano Calzavara, Tobias Urban, Dennis Tatang, Marius Steffens, and Ben Stock. 2021. Reining in the Web's Inconsistencies with Site Policy. In *Symposium on Network and Distributed System Security (NDSS)*. https://doi.org/10.14722/ndss.2021.23091

[11] Darion Cassel, Su-Chin Lin, Alessio Buraggina, William Wang, Andrew Zhang, Lujo Bauer, Hsu-Chun Hsiao, Limin Jia, and Timothy Libert. 2022. OmniCrawl: Comprehensive Measurement of Web Tracking With Real Desktop and Mobile Browsers. *Proceedings on Privacy Enhancing Technologies* 2, 1 (2022).

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

[13] Ha Dao and Kensuke Fukuda. 2021. Alternative to Third-Party Cookies: Investigating Persistent PII Leakage-Based Web Tracking. In *International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*. https://doi.org/10.1145/3485983.3494860

[14] Nurullah Demir, Matteo Große-Kampmann, Tobias Urban, Christian Wressnegger, Thorsten Holz, and Pohlmann Norbert. 2022. Reproducibility and Replicability of Web Measurement Studies. In *International Conference on World Wide Web (TheWebConf)*. https://doi.org/10.1145/3485447.3512214

[15] Nurullah Demir, Daniel Theis, Tobias Urban, and Norbert Pohlmann. 2022. Towards Understanding First-Party Cookie Tracking in the Field. In *Sicherheit, Schutz und Zuverlässigkeit (GI-Sicherheit)*.

[16] Nurullah Demir, Tobias Urban, Kevin Wittek, and Norbert Pohlmann. 2021. Our (in)Secure Web: Understanding Update Behavior of Websites and Its Impact on Security. In *Conference on Passive and Active Measurement (PAM)*. https://doi.org/10.1007/978-3-030-72582-2_5

[17] Clemens Deußer, Steffen Passmann, and Thorsten Strufe. 2020. Browsing Unicity: On the Limits of Anonymizing Web Tracking Data. In *IEEE Symposium on Security and Privacy (S&P)*. https://doi.org/10.1109/SP40000.2020.00018

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

[19] Steven Englehardt and Arvind Narayanan. 2016. Online Tracking: A 1-Million-Site Measurement and Analysis. In *ACM Conference on Computer and Communications Security (CCS)*. https://doi.org/10.1145/2976749.2978313

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

[21] Google Inc. 2022. BigQuery: Cloud Data Warehouse. https://cloud.google.com/bigquery.

[22] Muhammad Ikram, Rahat Masood, Gareth Tyson, Mohamed Ali Kaafar, Noha Loizon, and Roya Ensafi. 2019. The Chain of Implicit Trust: An Analysis of the Web Third-Party Resources Loading. In *International Conference on World Wide Web (WWW)*. https://doi.org/10.1145/3308558.3313521

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

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

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

[26] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Symposium on Network and Distributed System Security (NDSS)*. https://doi.org/10.14722/ndss.2019.23386 Generated on 03/04/2022. Available at https://tranco-list.eu/list/5KJN/1000000.

[27] Sourena Maroofi, Maciej Korczyński, and Andrzej Duda. 2020. Are You Human? Resilience of Phishing Detection to Evasion Techniques Based on Human Verification. In *ACM SIGCOMM Internet Measurement Conference (IMC)*. https://doi.org/10.1145/3419394.3423632

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

[29] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. 2016. Website Fingerprinting at Internet Scale. In *Symposium on Network and Distributed System Security (NDSS)*. https://doi.org/10.14722/ndss.2016.23477

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

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

[32] Alexander Sjösten, Peter Snyder, Antonio Pastor, Panagiotis Papadopoulos, and Benjamin Livshits. 2020. Filter List Generation for Underserved Regions. In *International Conference on World Wide Web (TheWebConf)*. https://doi.org/10.1145/3366423.3380239

[33] Martino Trevisan, Danilo Giordano, Idilio Drago, and Ali Safari Khatouni. 2021. Measuring HTTP/3: Adoption and Performance. In *Mediterranean Communication and Computer Networking Conference (MedComNet)*. https://doi.org/10.1109/MedComNet52149.2021.9501274

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

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

[36] Antoine Vastel, Walter Rudametkin, Romain Rouvoy, and Xavier Blanc. 2020. FP-Crawlers: Studying the Resilience of Browser Fingerprinting to Block Crawlers. In *MADWeb'20-NDSS Workshop on Measurements, Attacks, and Defenses for the Web*.

[37] Zhiju Yang and Chuan Yue. 2020. A Comparative Measurement Study of Web Tracking on Mobile and Desktop Environments. *Proceedings on Privacy Enhancing Technologies* 2020, 2 (2020).

[38] David Zeber, Sarah Bird, Camila Oliveira, Walter Rudametkin, Ilana Segall, Fredrik Wollsén, and Martin Lopatka. 2020. The Representativeness of Automated Web Crawls as a Surrogate for Human Browsing. In *International Conference on World Wide Web (TheWebConf)*. https://doi.org/10.1145/3366423.3380104

A AVAILABILITY OF DATA & CODE ARTIFACTS

To foster future research, we release our code, queries for the entire data processing pipeline and evaluation, and other supplementary information online at: https://github.com/internet-sicherheit/On-the-Similarity-of-Web-Measurements-Under-Different-Experimental-Setups. Furthermore, we provide the raw data collected during our experiments: https://doi.org/10.35097/1719/.

B ETHICS

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 loading 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 discussed aspects apply to all Web measurement studies, and our used practices are considered state-of-the-art and are accepted by our research community [11, 14, 24, 27, 31, 34, 38].

C MEASUREMENT FRAMEWORK

We develop our measurement framework in line with recent findings on how to run robust Web measurements [14, 24]. 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 [14]. 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 be visited 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 within 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* [21] database.

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 a German university network (*Westphalian University of Applied Sciences*), which might introduce some bias into our results. Furthermore, each VM runs 15 browser instances in parallel. We configured a timeout of 30 seconds for each page visit (similar to previous works [14, 34, 35]). Other works used a longer timeout (e.g., [20, 24]), but the effects of different timeouts have yet to be studied in detail and could be addressed in future work. We use a shorter timeout to allow fair scalability of our experiment, in which we visit over 1.7M pages. We do not perform any consequent experiments to understand the impact of the timeout on the results because previous work already highlighted such effects [14], 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, our study will provide a lower bound of the problem.

D VISUALIZATION OF OUR COMPARISON APPROACH

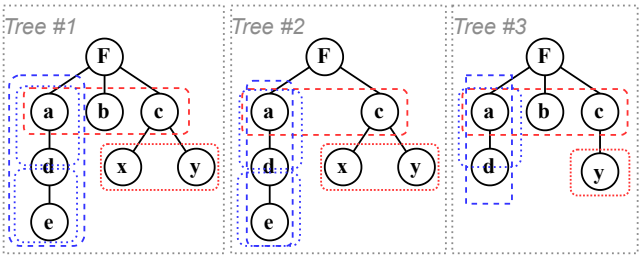

Figure 6: Depiction of our vertical (blue) and horizontal (red) comparison approach.

Fig. 6 provides an overview of our comparison approach. The blue boxes illustrate the vertical approach, and the red boxes illustrate the horizontal 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 more than one tree (dotted red squares—the figure shows only one example to increase readability). In the example, at depth one (red dashed squares), the given trees have a Jaccard index of

$$\frac{\frac{|\{a,b,c\}\cap\{a,c\}|}{|\{a,b,c\}\cup\{a,c\}|} + \frac{|\{a,b,c\}\cap\{a,b,c\}|}{|\{a,b,c\}\cup\{a,b,c\}|} + \frac{|\{a,c\}\cap\{a,b,c\}|}{|\{a,c\}\cup\{a,b,c\}|}}{3} = \frac{\frac{2}{3}+1+\frac{2}{3}}{3} = .77$$

and the Jaccard index for all nodes in all trees $\frac{\frac{6}{7}+\frac{5}{7}+\frac{5}{6}}{3} = .8$, while the index for the loading parent of node e (blue dashed square) is $\frac{1+0+0}{3} = .3$.

E NUMBER OF CHILDREN AT A SPECIFIC DEPTHS

Figure 8 provides an overview of the distribution in the number of children for each node based on their depth. A visual inspection of the plot shows that nodes on deeper levels in the trees include a higher number of (direct) children. This observation hints that website providers might not be aware of the inclusion of theses nodes, an observation that related work also made [34].

Table 7: 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

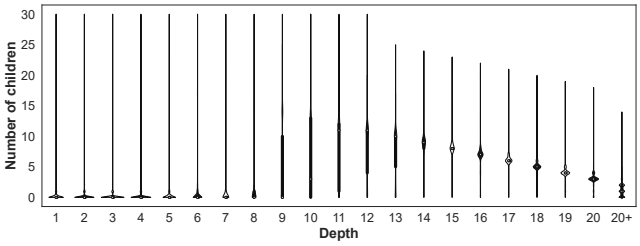

Figure 8: Number of children each node has at a specific depth. We capped the number of children at 30 and combined all depth levels deeper than 20 into one.

.

F UNDERSTANDING IMPLICATIONS OF SITE POPULARITY

Previous work has shown that a website's popularity impacts different phenomena [7, 34]. 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 7 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 Kruskal-Wallis test found statistical significance for 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 ($\epsilon^2 = .002$), meaning there is a statistically significant effect, but it is practically negligible. Thus, a site's rank does not impact the similarity when measuring it multiple times.

G SIMILARITY OF DIFFERENT RESOURCE TYPES

In Section 4.2, we show that the resource type can significantly affect the similarity of the observed trees. Figure 7 provides an overview of the average similarity of children and parent nodes of all resource types based on their depth in the trees. It shows 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); simultaneously, the similarity for some content types (e.g., script) changes drastically based on the observed depth.

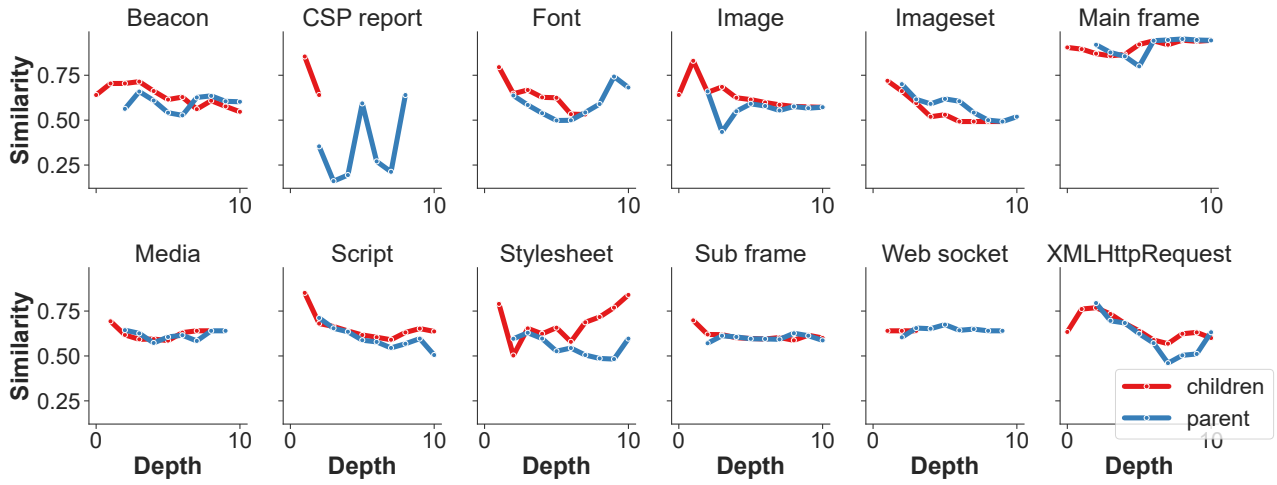

Figure 7: Average similarity of children and parent nodes for different resource types based on their tree depth.