Towards Realistic and Reproducible Web Crawl Measurements

Jordan Jueckstock†, Shaown Sarker†, Peter SnyderΔ, Aidan Beggs†,
Panagiotis Papadopoulos⋄, Matteo Varvello★, Ben LivshitsΔ, Alexandros Kapravelos†
† North Carolina State University, Δ Brave Software, ⋄ Telefonica Research, ★ Nokia Bell Labs

ABSTRACT

Accurate web measurement is critical for understanding and improving security and privacy online. Implicit in these measurements is the assumption that automated crawls generalize to the experiences of typical web users, despite significant anecdotal evidence to the contrary. Anecdotal evidence suggests that the web behaves differently when approached from well-known measurement endpoints, or with well-known measurement and automation frameworks, for reasons ranging from DDOS detection, hiding malicious behavior, or bot detection. This work improves the state of web privacy and security by investigating how, and in what ways, privacy and security measurements change when using typical web measurement tools, compared to measurement configurations intentionally designed to match "real" web users. We build a web measurement framework encompassing network endpoints and browser configurations ranging from off-the-shelf defaults commonly used in research studies to configurations more representative of typical web users, and we note the effect of realism factors on security and privacy relevant measurements when applied to the Tranco top 25k web domains. We find that web privacy and security measurements are significantly affected by measurement vantage point and browser configuration, and conclude that unless researchers carefully consider if and how their web measurement tools match real world users, the research community is likely systematically missing important signals. For example, we find that browser configuration alone can cause shifts in 19% of known ad and tracking domains encountered, and similarly affects the loading frequency of up to 10% of distinct families of JavaScript code units executed. We also find that choice of measurement network points have similar, though less dramatic, effects on privacy and security measurements. To aid the measurement replicability, and to aid future web research, we share our dataset and precise measurement configurations.

ACM Reference Format:
Jordan Jueckstock†, Shaown Sarker†, Peter SnyderΔ, Aidan Beggs†, Panagiotis Papadopoulos⋄, Matteo Varvello★, Ben LivshitsΔ, Alexandros Kapravelos†. 2021. Towards Realistic and Reproducible Web Crawl Measurements. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3442381.3450050

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450050

1 INTRODUCTION

Research into web security and privacy depends on web measurements for ground truth [18, 21]. Automated, web-scale crawls have been used to estimate user tracking granularity [10] and regulatory compliance [11] among other important questions. Ahmad *et al.* found nearly 16% of papers recently published across many top security, privacy, and network measurement venues relying at least in part on data collected via automated web crawls [7].

Implicit in most automated web measurement work is the assumption that the web encountered through automated measurement is the same as the web encountered by typical web users, or at least similar enough so that the findings for the former generalize to the experiences of the latter. However, as crucial as this assumption is to much security and privacy work, we find that this assumption has not been systematically studied or assessed.

More concerning, the related work that does exist suggests that the gap between the "measured" web and the "experienced" web may be large. For example, Zeber *et al.* [26] compared results between automated crawls from different network endpoints against anonymized browsing sessions provided by volunteers, they found some dramatic mismatches between the crawls and user sessions in key privacy metrics (*e.g.*, prominence of 3rd party domains contacted), but left unresolved the question of how much impact was attributable to sites deliberately discriminating against "unrealistic" clients.

Similar work by Ahmad *et al.* [7] assessed the impact of user agent choice on web crawl observations, from primitive headless agents like cURL to sophisticated full-browser frameworks like OpenWPM [10]. While these results again showed dramatic divergence in common security and privacy metrics, the authors' emphasis was on experiment reproducibility, and their analysis did not attempt to quantify the direct effects of bot detection/discrimination on results. Other recent publications document that many web sites perform dynamic bot detection [14] and that some malicious content actively evades visitors from non-residential networks [24].

This work aims to improve the state of web privacy and security by investigating how the web changes when observed with typical web measurement techniques, compared to measurement configurations carefully designed to closely match those of typical web users. More specifically, we measure how choices in browser configuration (BC) and network vantage point (VP) affect common privacy and security metrics. We design BCs and VPs to closely match those of typical web users and treat them as ground truth against which commonly used alternatives can be compared in a robustly controlled and repeated web crawl over the Tranco top 25K web domains [16].

This study considers three commonly used measurement VPs (*i.e.*, popular cloud provider, research university, residential ISP) and two common BCs (one with the default configuration of a popular browser automation framework, and the other configured

to more closely approximate a standard desktop browser). We treat the measurements taken from the *standard desktop browser* BC and *residential ISP* VP as "realistic", or ground truth (*i.e.*, the web as encountered by typical web users), and consider consistent differences in results between our ground truth configurations and the other VP and BC configurations to be a form of *measurement bias*. Higher levels of consistent difference between configurations are constitutes larger measurement bias, and thus a greater threat to validity to measurement studies employing unrealistic configurations. Our analyses ignore ephemeral outliers and identify data points showing *consistently significant* differences among VP and BC variants across *all* repeated crawls. We believe our approach establishes a lower bound on measurement bias (see Section 2.7) that can be expected from unrealistic web measurement methodology.

We find significant, and sometimes dramatic, differences in common privacy and security measurements attributable to VP and BC selection. A partial list of this work's findings include that certain commonly used measurement configurations introduce significant measurement bias regarding which domains are encountered, and how much traffic is sent to those domains. We find that measurements from cloud VP introduce higher measurement bias than other measured VPs. We also find that using non-realistic BCs introduces significant measurement bias regarding which well-known advertising and tracking domains are encountered; measurements taken with the default puppeteer[1] configuration, for example, miss up to 19% of realistic advertising and tracking domains encountered. We observe that non-realistic choices in BC and VP can introduce similar measurement bias into which JavaScript libraries are observed on the web. We also present case studies demonstrating that non-realistic measurement configurations cause different patterns in JS API calls.

Finally, we use our findings to provide recommendations for future web privacy and security research, to maximize "realism" in measurement results. We provide more detailed guidance and discussion in Section 4, but in summary, we conclude that researchers should avoid lowest-common-denominator crawlers, such as stock Puppeteer driving headless Chromium, when assessing real-world security and privacy concerns.

Contributions: Our core contributions include:

(1) **Comprehensive documentation and implementation details** of our synchronized parallel web crawl methodology, measuring how the privacy and security characteristics of the web change under different measurement configurations.
(2) The **complete dataset** generated by crawling the Tranco 25K top sites from 3 measurement vantage points, and under two representative browser configurations.
(3) **Conclusions and guidance on how future research should incorporate this work's findings** to improve how accurately findings from automated web measurements can generalize to real world, human browsing behavior.

2 METHODOLOGY

Our experiments center on simultaneous visits to top-ranked web sites by multiple clients differing in realism of network vantage point (VP) and browser configuration (BC), with controls in place to neutralize differences introduced by external factors such as available system resources, DNS resolution, and sources of programmatic entropy available to client-side JS scripts. Recent work [7] has documented an unfortunate propensity of authors employing web crawls to under-specify the design parameters of their crawls, frustrating reproduction of results. Here we specify and justify our crawl design criteria in reproducible detail.

2.1 Approach to Realism

As we attempt to measure the extent to which automated web measurements can be distorted by unrealistic (*i.e.*, non-human-like) crawlers, we face a challenge defining and deploying a "realistic" crawler. Ideally, we would compare a typical automated crawler directly against a live, human counterpart. Such an ideal experiment is impractical for several reasons: human volunteers do not scale well, and using real-world browsing data is fraught with ethical concerns, if it is even available at all. Our solution is to select VP and BC alternatives that can be reasonably ranked in order of *relative* realism based on known instances of adversarial response (*e.g.*, bot detection, malicious cloaking). We expect the differences observed (if any) between lower-realism and higher-realism crawlers, if all other factors are controlled, to provide a lower bound for expected differences between typical crawlers and actual human users.

2.2 Realism Variables

Two variables control the range of realism attempted by our clients: the network endpoint from which we visit pages (*vantage point*, or *VP*) and the browser settings employed (*browser configuration*, or *BC*). Each target URL (Section 2.4) is visited to produce a *page set*, the result of visiting that URL simultaneously from each distinct VP/BC pairing.

2.2.1 Vantage Point (VP). We collected data from three distinct, representative VPs: a major research *university* network, a nearby *residential* ISP network, and a popular *cloud* provider's network (Amazon AWS). The university and residential endpoints are co-located in the same city. The cloud endpoint was placed in the cloud provider's nearest available datacenter, which is within the same national border, in a neighboring province. The residential network provides the ostensible best-case in VP realism, as it is used exclusively for end-user activities. The university network combines both end-user and infrastructure activities; its realism is presumed to fall somewhere between the residential and cloud extremes. The cloud network provides an expected worst-case in VP realism as its typical use is for infrastructure rather than end-user network access. Connectivity via these endpoints is achieved via implementation details discussed in Section 2.5.

According to IP geolocation data provided by https://ifconfig.co, the university and residential endpoints were 6 km apart, while the university and cloud endpoints were 375 km apart. Naturally, the distance separating the cloud endpoint from its counterparts raises concerns about the effect of geo-targeted web content. We address this concern in the discussion of our analysis approach (Section 3).

2.2.2 Browser Configuration (BC). We crawl using two variants of Puppeteer[2] controlling Chromium 80: a lower-realism *naive* variant and a higher-realism *stealth* variant. Relative realism is inferred

[1] https://github.com/puppeteer/puppeteer/

[2] https://github.com/puppeteer/puppeteer/

from the ongoing arms race between developers of bots (automatic, non-human user agents) and bot detectors. The naive BC, running Chromium in headless mode using stock Puppeteer, is easy to detect as a bot thanks to identifying quirks [4] of Chrome's headless mode, (*e.g.*, "ChromeHeadless" in the User Agent string). The stealth BC presents a harder target by running the browser in non-headless mode and using a community-provided stealth plugin for Puppeteer that adds bot-detection countermeasures such as spoofing available media codecs to better match consumer devices and suppressing the tell-tale `Navigator.webdriver` attribute. The existence and continued maintenance of the stealth plugin, an explicit workaround for bot detection of headless Puppeteer crawlers, indicates that there exists some population of content in the wild for which our naive BC will be considered "unrealistic" and our stealth BC "realistic."

2.2.3 Summary. Each of our page sets comprises 6 synchronized parallel visits to the same URL, one for each combination of our 3 network VPs and 2 BCs representing a range of relative realism levels.

2.3 Control Constants

Our analysis of results across VP/BC depends on eliminating as many sources of irrelevant differences across clients as possible. To this end we aggressively homogenize all readily controllable aspects of our crawls across all clients.

2.3.1 Workflow & Timeouts. All page visits follow the same workflow and employ the same timeout limits for each phase. First, the browser is launched with a clean user profile (*i.e.*, no cookies or cached content), instrumentation callbacks are established, and the browser is navigated to the target URL. If this initial navigation fails to successfully fetch an HTML document within the *navigation timeout* of $T_N = 30$ seconds, the page visit is aborted. Otherwise, the browser is left idle, running JS code and firing timers as needed, for the *loiter timeout* of $T_L = 15$ seconds. Once the loiter time is expended, the crawler begins to "tear down" the visit by first capturing a number of page artifacts (such as final DOM HTML and screenshot). If this "tear down" process exceeds the *watchdog timeout* limit of $T_D = 15$ seconds, the page visit is aborted. (All non-aborted page navigations are considered successful, even if they never fire the official "load" event.) The theoretical maximum time taken per page, then, is 60 seconds. We discuss selection of these parameter values in Section 2.6

2.3.2 DNS Resolution. We configured all browsers at all endpoints to resolve DNS names using CloudFlare's popular `1.1.1.1` resolver network [1]. Our experiment is not designed to measure the impact of DNS resolution on web content, so we did not perform A/B crawls with and without CloudFlare's DNS service. Rather, our goal is to reduce potential noise caused by using different DNS servers from different providers with completely different priorities and quality of service.

2.3.3 JS Entropy Sources. Dynamic resource loading triggered by JS code has been known to rely on sources of randomness available to JS programmers, such as the `Math.random` API, or on the current timestamp as returned (with millisecond granularity) by `new Date()`. Whenever a new frame is created, our crawler pre-loads its execution context with a JS polyfill from the Google Catapult project's Web Page Replay framework [3] that provides deterministic alternatives to these APIs.

2.3.4 Bandwidth/Latency. To compensate for differences in available bandwidth and typical latency between our VPs, we used Chromium's network throttling support to limit maximum throughput and minimum request latency to the lowest-common-denominator in our setup. Unsurprisingly, our network bottleneck was the **residential** VP, equipped with asymmetric bandwidth (200 Mbps down, 10 Mbps up) which had to be shared with two residents compelled to work from home thanks to the COVID-19 pandemic. We set bandwidth limits arithmetically, by dividing 50% of available residential bandwidth (each direction) among the workers deployed there and setting identical limits on all other workers. We measured the highest round-trip time for a simple HTTP request time experiment conducted through each VP and used the maximum of **64ms** (from residential) as the minimum latency for all workers.

2.3.5 Summary. All anticipated entropy sources are held as constant as is practical across all visits: workflow timeouts, DNS resolution (via CloudFlare), JS entropy sources, and lowest-common-denominator bandwidth/latency throttling.

2.4 Web Site Selection

We visit the top 25,000 web domains as ranked by the Tranco list of top sites [16] (snapshot 77PX). In keeping with our focus on web security and privacy measurements, our interest is primarily 3rd-party infrastructure content such as advertisement frameworks, trackers, and analytics scripts rather than 1st-party application content or behavior, so we do not expend resources crawling recursively into a website's contents beyond the "landing page" provided by navigating to the domain name itself as an HTTP URL.

Much web content is inherently dynamic [5] (*e.g.*, news headlines, advertisements) or even personalized (*e.g.*, recommended content, advertisements). Furthermore, the web depends on a strictly-best-effort Internet, where ephemeral connectivity issues are common. Such ephemeral noise threatens our ability to isolate meaningful differences across endpoints. In addition to all the controls enumerated above, we combat such noise with *repetition*, visiting our itinerary 3 times and factoring consistency across repetitions into our analysis. Repetition count is not a well standardized parameter of web crawl methodologies. If mentioned at all, it is typically justified in relation to a particular metric or analysis technique [7, 22]. We chose 3 repetitions pragmatically: it provides some robustness against temporary connectivity issues and provides measurement of stability in observations across crawls while retaining modest resource overhead.

Since some environmental factors outside our control (*e.g.*, diurnal activity patterns affecting network load) directly relate to time, and since even responsible web crawling at slow rates might well result in IP blacklisting over the course of a week-long crawl, we decouple website popularity from time elapsed during the experiment by *randomizing* the order of domains visited at the start of each crawl repetition. This shuffling prevents diurnal patterns from coinciding with domain spacing in our crawl and gives us some confidence that any rank-based metrics used in our analysis are not accidental proxies for time-of-day or other temporal patterns.

[3] https://chromium.googlesource.com/catapult/

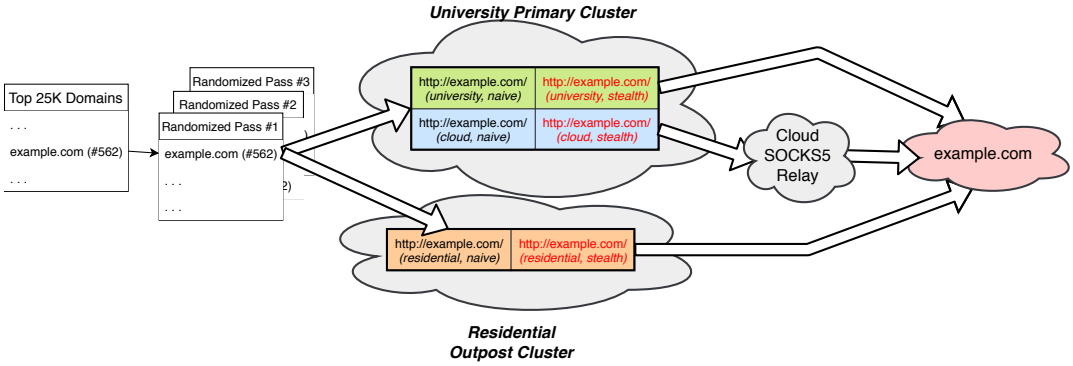

Figure 1: Workflow from Domain List to Target Server

2.4.1 Summary. We visit the Tranco top 25,000 domains in a series of 3 independent crawls. We randomize the order of site visits within each crawl to decouple our results from potential time-based confounding factors.

2.5 Implementation Details

Figure 1 illustrates the high-level design of our crawling experiment and hints at some of the implementation and infrastructure details briefly discussed here.

2.5.1 Infrastructure. All *university* and *cloud* visits were hosted on an 8-node Kubernetes cluster, comprising 352 total CPUs and 1.5TiB total RAM. Cloud visits were proxied through our Amazon AWS endpoint using Go Simple Tunnel's [3] SOCKS5-over-KCP low-latency encrypted transport mode. Given the asymmetric bandwidth available from the residential ISP, described in Section 2.2, we could not tunnel crawls through this endpoint, as the tunnel would be effectively throttled by the crippled upstream rate. Instead, we placed a single-node (16 CPUs, 32GiB RAM) Kubernetes cluster at the residential endpoint. Workers on this *outpost* cluster handled all visits from the residential VP. All workers in the experiment (in both clusters) were configured with CPU and memory limits derived to strictly prevent saturation of the outpost's limited resources. Bulk data (e.g., HTTP response bodies, VisibleV8 trace logs) was stored in a local MongoDB server running alongside each cluster. Post-processed summary data was stored in a single PostgreSQL server colocated with the primary cluster: outpost post-processing jobs communicated with this server via persistent SSH tunnel.

2.5.2 DNS Customization. The Chromium browser does not provide runtime options to select a custom DNS resolver. Our workaround was to configure all Chromium instances, not just those visiting via the cloud endpoint, to use a *local* SOCKS5 proxy configured to use our desired remote DNS server for name resolution. This approach gave us easy control over DNS server selection without altering the originating IP address of the request, and had the additional benefit of normalizing connection overhead, proxy latency, and Chrome error reporting between the local and cloud endpoints.

2.5.3 Synchronization. Even with careful resource tuning and throttling, page sets will not remain synchronized across a multi-day crawl experiment without help. Unsynchronized tests over the top 1K sites revealed that page visits from the same set would be spaced far apart soon after the experiment began: e.g., by the time 400 domains had been processed, the residential visits were already over 30 minutes behind their cloud and university counterparts (which were less than a minute apart), despite uniform CPU and memory limits and no sign of network bandwidth saturation at any VP. To provide maximum comparability across control variables, we set out to synchronize visit starts within page sets to be nearly instantaneous.

Workers in both clusters pulled page visit jobs from a single Redis-backed work queue hosted in the primary cluster. Outpost workers accessed this Redis server via a persistent SSH tunnel between the clusters. We augmented the off-the-shelf work queue logic with a custom *synchronization barrier* implemented using atomic counters and pub/sub notifications provided by the same central Redis server. Under this scheme, each job in a page set is tagged with a shared *sync tag* which serves as a Redis *key* for storing an atomic counter (initially 0). After pulling a job from the queue but before starting the visit, workers *subscribe* to notifications for that sync tag, *atomically increment* the sync counter and, if the returned value matches the expected total count (*e.g.*, 6), *publish* a notification to release all workers waiting on that tag. With this implementation in place, 99.8% of all page sets in our primary experiment saw all 6 visits launched within a 1 second window, with a mean launch window of 91ms.

2.5.4 Summary. We split our infrastructure into a *primary* and an *outpost* cluster to work around asymmetric bandwidth limits for the residential VP. DNS resolution customization is controlled via *local* and *remote SOCKS5 proxies*, simplifying implementation and unifying behavior and reporting across all variants. Work is distributed and synchronized across the clusters via a central Redis server using standard work-queuing and custom *barrier synchronization* techniques.

2.6 Precautions & Pilot Experiments

As there is no universal "ground truth" for web crawl data collection, we can validate our system only in a precautionary, best-effort sense. We list here predictable threats to validity which we considered and mitigated, along with experimental confirmation of reasonable results.

Is the navigation timeout $T_N = 30s$ reasonable?: We believe so, for two reasons: the 30 second timeout is comparable to timeouts in similar work [22], and it is longer than what we expect a typical user to tolerate, based both on widely agreed-upon web user experience guidelines [2] and past user behavioral studies [19].

Is the loiter timeout $T_L = 15s$ adequate to allow full page load before shutting down?: Yes. We performed a pilot experiment over the Tranco top 1K testing loiter times ranging from 15s to 60s in 5s increments and found *no* variation in how many pages achieved a full "page load" event. We did not test loiter times below 15s as the distribution of successful page load times indicated 15s to be the minimum reasonable lower bound.

Do the network bandwidth and latency throttling controls distort page performance?: No. We tested the Tranco top 1K with and without bandwidth/latency throttling and found essentially no difference in error rate and other core statistics. The results were in fact so similar we were tempted to eliminate the throttling from the experiment controls. But given past experience suggesting that top sites behave better than average, we left the controls enabled in the event that lower-ranked sites generate load such that the mismatch in available bandwidth between clusters might harm the comparability of results.

Are the limited resources available at the residential outpost able to keep pace with their beefier counterparts?: Yes. We set CPU, memory, and bandwidth limits on all workers to lock total use of system resource below potential saturation for the lowest common denominator environment (the residential outpost). Pilot tests during heavy residential network usage (e.g., video conferencing) revealed neither impact to our collection speed or success nor degradation observable by the residents. During this test we specifically monitored the upstream bandwidth used by post-processing when shipping data back to Postgres via SSH tunnel and found it well-constrained below a peak rate of 1.5Mbps without any specific limits or throttling being applied.

2.6.1 Primary Experiment Consistency. The primary experiment ran from 30 April to 9 May (2020). We queued a total of 450,000 page visits, of which 449,936 (99.99%) completed without fatal error. Of these completed pages, 375,246 (83.40%) were completely successful and 74,690 (16.60%) experienced some level of failure. Note that we are extremely conservative in labelling failure, including pages that loaded and collected content successfully but which failed to shut down collection in a timely manner and were thus forcibly aborted late in the page visit workflow. The reported failure rate is thus a lower bound on the number of visits producing useful data for analysis.

We confirmed that our collection was free of any unexpected patterns in failure rates relating to Tranco rank, time of day, or day of experiment. The Tranco rank independence reassured us that our pilot experiments focused on the top 1K applied reasonably to the rest of the crawl. The time of day independence reassured us that our residential outpost workers were coexisting peacefully with the residential traffic. And the day of experiment independence reassured us that our endpoints were not subjected to effective blacklisting or other progressive service degradation over the course of the experiment, as confirmed independently by reputation monitoring provided by https://hetrixtools.com/.

2.6.2 Summary. We justify our choice of timeouts from prior practice and specific testing. We verify that our results are not contaminated with noise from resource saturation within either cluster or from resource mismatch between clusters. We find that errors do not appear correlated to potential time- or rank-based confounding factors, implying that our observations are consistent and reasonably robust.

2.7 Quantifying Measurement Bias

Our analysis depends on quantifying how much crawlers, differing only in relative realism, record consistently, significantly different web measurement results. A specific measurement that consistently, significantly differs across a realism variable like VP or BC constitutes **measurement bias**.

2.7.1 Bias Scores. To facilitate comparing and reasoning about bias, we quantify it to produce a concrete score. We begin by aggregating an additive *metric* (*e.g.*, total requests) grouped by an *entity* (*e.g.*, eTLD+1 domain of an HTTP request URL) and a *control* variable (*e.g.*, VP or BC). We then compute each entity's *bias scores*, one for each distinct combination of control variables (*e.g.*, stealth-*vs.*-naive, or residential-*vs.*-cloud). For a pair of metrics a and b, the score formula is $R(\log_2 a - \log_2 b)$, where $R(x)$ is the common integer rounding function (rounding away-from-0 at .5). Using differences of logarithms provides more descriptive power than a simple count difference while avoiding the extreme outliers likely when using ratios. A stealth-*vs.*-naive score of 2.0 for the domain `example.com`, for instance, shows that sites sent 4× as many 3rd party requests to `example.com` during stealth crawls than on naive crawls. A score of −1.0 would indicate 2× as many requests on naive crawls compared to stealth crawls. A score close to 0 (*i.e.*, the majority of scores in practice) indicates insignificant bias across our experiments for that domain.

2.7.2 Bias Consistency. Transient outliers are eliminated by independently scoring results from each of our 3 crawl repetitions and keeping only the entities (e.g., the 3rd party domains or JS script families) found in all 3 measurement sets. This intersection constructs a *synthetic score set*, where each datapoint is computed as the *median* of corresponding bias scores from the 3 measurement sets.

Intersecting the measurement sets also provides insight into how consistent the bias scores are across crawl repetitions. For each median bias score recorded in the synthetic score set, we keep the *count* of distinct bias scores from which the median was picked, a value in the range $[1,n]$ where n is the number of measurement sets ($n = 3$ in our experiment). Given a mean score-count C_m, a consistency score C_s can be computed, ranging from 0 (total inconsistency) to 1 (total consistency).

$$C_s = \frac{(n-1)-(C_m-1)}{n-1}$$

2.7.3 Summary. We quantify magnitude and consistency scores of bias for identical datapoints across endpoints. Visualizations of these metrics are introduced and explained in Section 3.2.

3 RESULTS

We analyze the data collected via the experiments described in Section 2 to assess the impact of crawler realism on simple, quantifiable metrics such as volume of HTTP requests to 3rd party domains. We apply our quantified measurement bias (Section 2.7) methodology to progressively finer-grained breakdowns of the data collected in our crawls: first using all 3rd party HTTP requests, then such requests flagged by ad and tracker filter lists, then flagged requests divided by Same Origin Policy (SOP) isolation context, and finally considering

VP/BC	# of Refuseniks
Cloud	72
Naive	69
Stealth	30
Residential	11
University	2

Table 1: Some "refusenik" sites always fail navigation from a single configuration but not its' complements

some of the content loaded itself (*i.e.*, JS script bodies). The results show a significant number of 3rd party domains and JS script families exhibiting consistently mismatched results across VP/BC, implying lack of crawler realism can significantly bias crawl results.

3.1 Refusenik Sites

We identified a small but impressive collection of "refuseniks": sites that *always* failed to load from a particular vantage point (VP) or browser configuration (BC) but which *never* failed to load from complementary configurations (naive vs. stealth, for instance). The total number and category of these sites is provided in Table 1. The largest categories for which service was refused (cloud for VP, naive for BC) are intuitive, confirming expectations that some sites aggressively block probable bots or crawlers altogether. The residential and university share is small enough to be within the realm of accident, but the number of stealth-refusing sites is puzzling. One possible explanation is that since each visit involves 2 parallel requests (one naive, one stealth) from each endpoint IP, with high likelihood that the stealthy request will be slightly later than the naive request (as headless browsers have lower startup overhead), the stealth requests are more likely to run afoul of over-aggressive per-IP rate limiting. We do not investigate farther, as the number of refuseniks is too low to impact our other measurements.

3.2 Volume Biases in HTTP Traffic

Defining and Visualizing Request Bias: The volume of HTTP requests sent during a page visit, broken down by the target domain (specifically the *eTLD+1*, or the public DNS suffix plus one additional label), is a useful metric for quickly gauging both the richness of a page's content and the potential advertising/tracking privacy footprint of a visit to that page. When applied to the volume of HTTP requests sent to 3rd party domains during page visits, the measurement bias methodology described in Section 2.7 identifies domains which (1) are contacted on each of our 3 experiments and which (2) serve significantly different levels of traffic to different VP/BC crawlers (*i.e.*, are biased for/against a given configuration). Bias visualizations such as Figure 2 plot the distribution of bias scores along with total percentages of eTLD+1 domains having bias scores < 0, $= 0$, and > 0 for each curve, along with the percentage of total HTTP requests associated with that set of contexts, and the corresponding bias consistency scores. Logarithmic scale is used on the Y axis to keep the curves meaningful, as the central 0 column (*i.e.*, the non-biased contexts) typically overpowers the tails containing the significantly biased entities. E.g., in Figure 2, the "R/C" curve plots bias scores for request volume per domain compared between residential (R) and cloud (C) VPs. 3.5% of the consistently-present

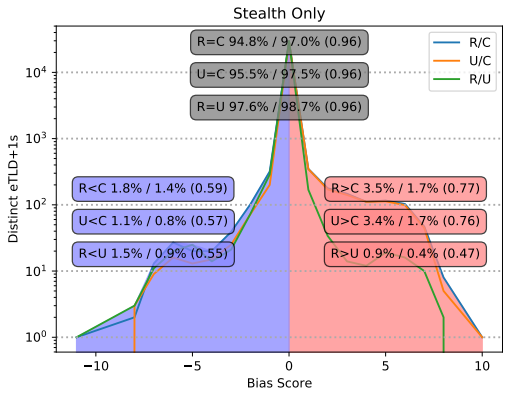

Figure 2: Distributions of cross-VP request volume bias by 3rd-party domains (stealth BC only); nearly twice as many domains consistently favor residential VP over the cloud VP as *vice versa* (3.5% > 1.8%).

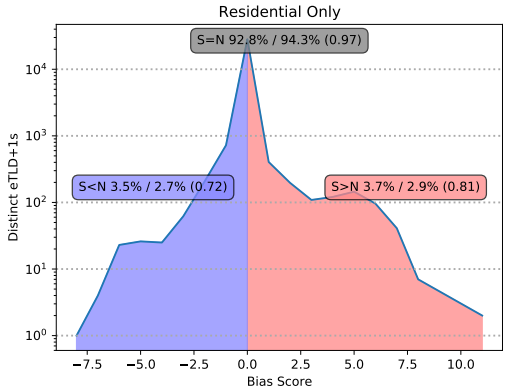

Figure 3: Distribution of stealth-*vs.*-naive traffic volume bias scores for 3rd-party domains (residential VP only); more symmetric than its cross-VP counterparts.

domains exhibited bias scores > 0 (*i.e.*, pro-residential) and 1.8% bias scores < 0 (*i.e.*, pro-cloud). The pro-residential domains account for 1.7% of all requests observed, while their pro-cloud counterparts account for 1.4% of all requests. The pro-residential domain set score noticeably higher in consistency (0.77 *vs.* 0.59).

3.2.1 Measuring Request Bias. We find measurement bias in the volume of all requests per target eTLD+1 across both VP and BC (Figures 2 & 3). Some of the pro-cloud biased domains probably serve ad content that geo-targets the cloud endpoint's location. But the asymmetry of the anti-cloud bias (*i.e.*, about twice as many domains show pro-residential bias as show pro-cloud bias) argues against geo-targeting, which ought to have roughly symmetric effect between two regions, as the defining factor. Cross-BC measurement bias has no obvious geo-targeting component, so the near parity in total domains showing pro-naive *vs.* pro-stealth bias is somewhat surprising. The opposing sides of the BC bias curves do show subtly asymmetric shape (which we note is strongest on the most realistic VP, residential): pro-stealth bias is concentrated among a smaller number of domains with higher bias scores, while pro-naive bias is concentrated among more domains with less pronounced bias scores.

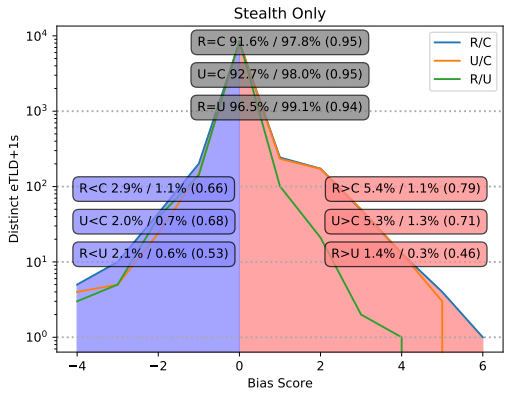

Figure 4: Distributions of cross-VP ad/tracker request volume bias by 3rd-party domains (stealth BC only); little change from global cross-VP distributions.

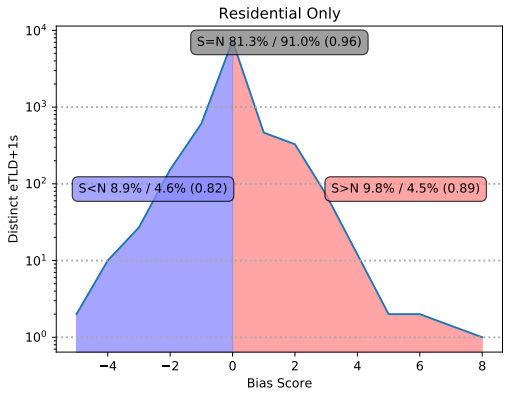

Figure 5: Distribution of stealth-*vs.*-naive ad/tracker traffic volume bias scores for 3rd-party domains (residential VP only); BC bias is more common among these domains than the global population.

We find that domains showing significant measurement bias account for reasonable shares of overall request volume. The set of consistently-present (*i.e.*, not ephemeral, or seen in only one sub-experiment) domains visualized in Figures 2 & 3 represents 92-93% of all domains encountered in any of our sub-experiments, and these consistent domains account for 98-99% of all requests recorded. These numbers demonstrate that our bias set intersection successfully (Section 2.7) captures the core of domains that account for the overwhelming majority of traffic while eliminating transient red herrings. The share of total requests associated with the biased domains for each curve (the second percentage listed for each curve) is consistently lower than the corresponding share of domains, but never so much so as to be trivial. These domains showing cross-VP and cross-BP traffic volume measurement bias are clearly not dominant, high-traffic providers, but in aggregate they account for non-trivial volumes of traffic, especially considering cross-BC bias.

We believe bot-detection, whether to shield proprietary data from scrapers or to avoid useless/fraudulent advertising impressions, to be a significant source of the observed measurement bias, as predicted by our realism-by-proxy design argument (Section 2.1). Manual inspection of highly-Tranco-ranked domains present at both extremes of the VP/BC bias distribution curves generally supports this theory. The intersection of pro-stealth and pro-residential bias outlier domains ordered by Tranco rank revealed a number of high-profile brands and content providers within the top 25: `usnews.com`, `accuweather.com`, `lego.com`, `lowes.com`, `dhl.com`, `hotels.com`, `expedia.com`, and `ti.com`. Only a few similarly high-profile brands appear in the top-ranked 25 domains from the complementary intersection of pro-naive, pro-cloud bias outliers, e.g. `amazon.de` and `audible.com`. The notion that premium content providers, especially those serving dynamic and potentially proprietary price data, would show significant anti-bot bias is hardly surprising, nor is it of great interest to security and privacy researchers. A more useful question is whether known ad and tracking traffic exhibits similar biases.

3.2.2 Request Bias by Known Ad/Tracker Domains. Of more concern to security and privacy researchers, we find traffic volume measurement bias patterns more pronounced among requests flagged by one or both of the popular, community-maintained EasyList (EL) and EasyPrivacy (EP) filter lists. In our experiment roughly **a third of all recorded requests** matched blocking rules in one or both of these lists; see Table 2. Figures 4 & 5 illustrate the shifts in measurement bias distributions across VP (using stealth BC) and bias across BC (using residential VP). The maximum bias score values shrink, but the total share of biased domains increases to nearly 20% of those contacted in all crawls. Furthermore, the consistency of cross-BC bias distributions for flagged requests increases over that of the global cross-BC distributions ($0.89 > 0.81$ and $0.82 > 0.72$).

Blocklists Involved	All Requests	Main Frame	1st Party Sub-frames	3rd Party Sub-frames
-/-	23,970,789	21,051,929	819,194	2,043,325
-/EP	5,470,406	3,855,858	227,974	1,333,700
EL/-	5,032,267	2,085,177	647,012	2,197,150
EL/EP	1,142,609	581,767	71,877	475,750

Table 2: Total HTTP requests by EasyList/EasyPrivacy match and frame context

As when considering all requests, we find biased domains to account for a respectable share of all EL/EP-flagged requests. The set of all consistently-present domains visualized in Figures 4 & 5 accounts for only 85-89% of all domains associated with any EL/EP requests, but in every case these consistent domains account for over 99% of all EL/EP requests. Once again, the filtering effect of intersecting results across multiple crawls eliminates significant chaff from the results.

Domains showing high bias cross-VP saw both a relative domain share *increase* and a relative request volume share *decrease*. But domains showing high bias cross-BC showed a more intuitive correlation between request volume and domain share, with both increasing significantly over the all-requests distributions. Again, these domains clearly do not dominate traffic volume, but particularly in the case of BC bias outliers, they account for a non-trivial amount of requests flagged by our filter lists.

When considering only likely advertising and tracker content, the puzzle of the BC curves' near symmetry is amplified. It makes sense for advertisers or trackers to show pro-stealth bias as a result of detecting and avoiding an obviously automated browser. But the nearly equal share of domains showing apparent pro-naive bias is counter-intuitive. Some portion of this activity, especially that

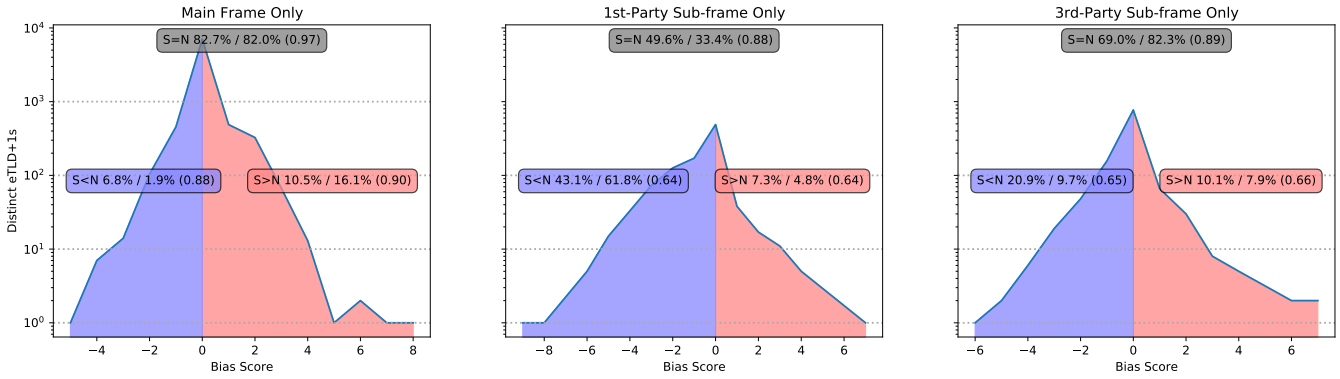

Figure 6: Distributions of stealth-*vs.*-naive ad/tracker traffic volume bias scores for 3rd-party domains (residential VP only) broken down by browser frame context; sub-frames show radically different (and less intuitive) BC bias distributions than main frames.

with low bias scores, is probably still related to defensive analytic behavior triggered by the presence of a headless client. But the presence of more extreme bias scores do not fit this explanation as well, given that such fingerprinting and reporting behavior ought to require significantly less request volume than serving typical ads or monitoring user activity over time.

A partial explanation may be displacement of content providers that more aggressively block bots by those which do not during naive crawls. There is a statistically significant (though modest in effect size) difference in distribution of Tranco ranks between the pro-stealth and pro-naive sets of biased domains across all VPs, with the median rank of the pro-stealth bias set being consistently higher (*i.e.*, a lower number) than its pro-naive counterpart. E.g., considering only residential VP data, the median naive rank = 18,360.0, the median stealth rank = 12,198.5, and a Mann-Whitney one-tailed test finds significant difference with $P < 0.001$[4]. Other VPs given slightly different values but show the same direction and scale of gap in median ranks, the same order of magnitude for P, and near-identical effect size (0.60). The modest effect size is unsurprising given the significant overlap in distributions, but the bias is unmistakable and consistent. We note that the rank skew between stealth and naive biased-domain sets is likewise visible (and statistically significant to the same level, albeit with slightly smaller effect size) in the bias sets derived from *all*, as opposed to only requests flagged by filter lists.

3.2.3 Request Bias & Frame Context. Continuing to consider only requests tagged by filter lists for ad and tracker content, we find more dramatic and surprising shifts in measurement bias when breaking down comparisons by browser frame/security contexts and security origins. [5] Here we consider only cross-BC measurement bias within the residential VP, as shown in Figure 6. Cross-VP measurement bias patterns are unchanged from previous analyses and are not discussed further. Note that these traffic share percentages are *not* restricted to a particular frame context but are computed globally for all requests tagged by our filter lists.

Sub-frames, of both first and third party domain origin, account for only 22% of all requests but 44% of requests matched by EasyList or EasyPrivacy rules. Unsurprisingly given their overall traffic share (Table 2), main frame bias score distributions closely follow the overall distributions. Sub-frame BC bias scores, however, both grow in overall share and swing counter-intuitively to the pro-naive side, probably in part because the much smaller share of traffic being considered is more readily influenced by popular outliers.

The pattern of lower Tranco ranks (higher popularity) found in the pro-stealth distributions *vs.* their pro-naive counterparts, present in both previous breakdowns, evaporates for sub-frames. Many factors contribute: the relatively small set of domains involved, the extreme mismatch in set size, and the fact that, on manual inspection, we found a fair number of extremely high-ranked domains had crept into the pro-naive bias set (*e.g.*, 4 Google domains, including `google.com`, in the top 10 pro-naive outliers). The presence of "heavy-hitter" domains is a first in our outlier analysis so far, and is underscored by traffic share analysis.

The set of domains consistent across all crawls for main frame EL/EP requests comprised 83-84% of all domains associated with any EL/EP request, much like the previous all-frames breakdown. As seen above, though, the sub-frames are different beasts altogether. The total set of consistent domains considered here comprises only 9-11% of all EL/EP-associated domains, but these account for an impressive 89-90% of all EL/EP flagged requests (obviously, when considering all frames). That large share of traffic going to a small subset of domains is not surprising when considering the presence of dominant players like Google. What is surprising is how decisively skewed this bias distribution is to the pro-naive side. We suspected that the presence of Google and other top-tier players in this small pocket of bias might be related to CAPTCHA deployment, but a search of the 31,787 distinct *stemmed* URLs (consisting only of hostname eTLD+1 and path component, sans query string) within this context and BC yielded only a single hit on any obvious variant of the word "CAPTCHA," in a single URL requested by a single site, once per visit. Of course, there is no reason to believe adversarial content will

[4] $U = 427615.5$, $n_1 = 801$, $n_2 = 882$

[5] All frames are associated with a *security origin* URL scheme, hostname, and port. It may match the origin of the main document (a 1st party frame) or not (a 3rd party frame), in which case the browser's Same Origin Policy (SOP) will restrict its access to the main frame's contents. 3rd party frames commonly host advertising and tracking content.

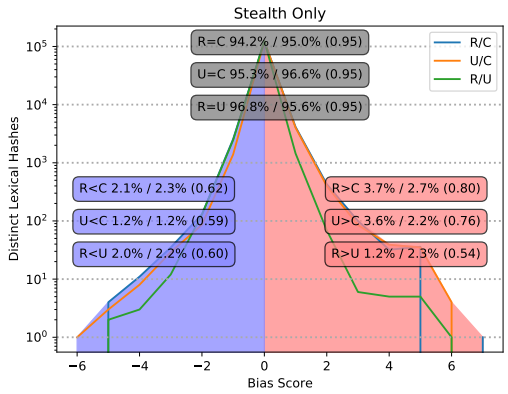

Figure 7: Distribution of cross-VP execution frequency bias for families of JS code (stealth BC only).

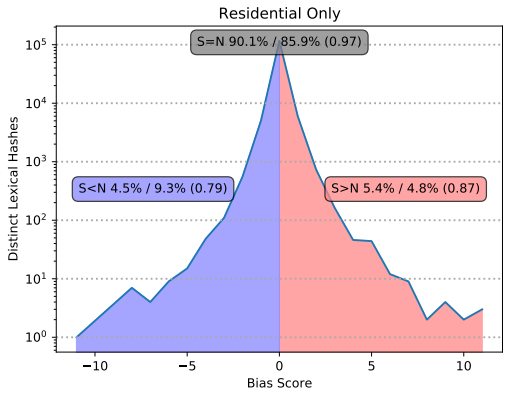

Figure 8: Distribution of stealth-*vs.*-naive execution frequency bias for families of JS code (residential VP only).

always advertise itself as such in domain names or URL paths, and it remains plausible that at least some of this pro-naive phenomenon is related to active adversarial response to suspected bots.

3.2.4 Summary. Around 5% of content-providing domains show significant measurement bias across VP, clearly favoring non-cloud endpoints. From our most realistic VP (residential), measurement bias across BC among HTTP traffic domains is more prominent, accounting for over 7% of domains and over 5% of total HTTP traffic volume. Request volume measurement bias becomes notably more pronounced among domains flagged by filter lists as serving ads and trackers, with nearly 20% of domains' traffic strongly correlated to choice of BC.

3.3 Content-Level Biases in JavaScript

3.3.1 Biases in Scripts Loaded. We find that the loading and execution of JS script families shows measurement bias patterns comparable to domain bias in requests. We define a "script family" to be a set of JS scripts observed loading and executing by the VisibleV8 [14] JS API tracing system that all share a common *lexical hash*. We compute lexical hashes by tokenizing all JS scripts using the industry standard Esprima JS parser and computing the SHA256 hash of the resulting sequence of token *type names*. Lexical hashes thus ignore variance in whitespace, comments, identifiers, and atomic values like number or string literals. We computed 258,236 total distinct lexical hashes from 1,517,281 total distinct scripts, not including 4,364 distinct scripts that failed tokenization because of syntactic irregularities (such scripts are excluded from lexical hash based analysis). To facilitate correlating script loading and HTTP traffic patterns, we consider only scripts loaded via URL (as opposed to `eval` or similar means). For reference, 87.9% of script families we observed were loaded at least once via a script URL (either the source of a `<script>` tag or the URL of an HTML document statically embedding JS code).

Measurement bias visualizations for script loading (Figures 7 and 8) show bias score distributions for distinct lexical hashes of script bodies (*i.e.*, "script families"). Figures include the percentage of total script loads/executions associated with each subset of script families (*i.e.*, "execution share"). Script loading measurement bias across VP (considering only stealthy BC; Figure 7) reveal approximately 5% of the stable script family population showing persistent bias for/against a given VP. Execution shares are closely

Script Domain Bias Sets	General Domain Bias Sets <	=	>	Script Domain Bias Sets
R<C	**11.1%**	87.1%	*1.8%*	
	2.1%	66.6%	**31.2%**	R>C
U<C	**14.1%**	84.1%	*1.8%*	
	2.2%	66.0%	**31.8%**	U>C
R<U	**9.6%**	89.5%	*0.9%*	
	1.2%	85.5%	**13.3%**	R>U
S<N	**12.3%**	82.9%	*3.9%*	
	2.8%	71.0%	**24.9%**	S>N

Table 3: Many domains showing no overall request volume bias serve script content with distinct VP/BC biases.

aligned with overall script family population share (e.g., 2.1% of script families show pro-cloud VP bias, and these account for 2.3% of observed script executions). The clearest direction of VP bias is, unsurprisingly, against the cloud endpoint (population shares of 3.7% *vs.* 2.1% and 3.6% *vs.* 1.2% for the R/C and U/C curves, respectively). Bias across BC (considering only residential VP; Figure 8) is twice as pronounced, with about 10% of the script family population showing consistent BC preference. Script loading BC bias appears fairly symmetric, as with request/domain BC bias. One curious quirk appears in BC script loading bias: the set of unbiased script families is extremely stable (0.97 consistency score) but accounts for *less* execution share (85.9%) than population share (90.1%). The "missing" execution share appears to be mostly lost to the pro-naive bias set (9.3% execution share vs. 4.5% population share). This imbalanced distribution is consistent with the likely scenario of servers engaging in selective delivery of a relatively small population of bot-detection scripts to obviously automated (*i.e.*, headless) clients.

When we correlate script loading biases with overall HTTP request volume biases, linked by source eTLD+1, we see that even domains showing no significant request volume bias still show measurement bias in the script content they serve. Each row of Table 3 shows the intersection of a given script loading bias set (*e.g.*, "R<C", or favoring-cloud-over-residential) with the general HTTP request volume bias scores of the eTLD+1s from which the scripts were loaded. In each case, a majority of domains serving execution-biased scripts do not show significant bias in overall request volume (*i.e.*, the majority share is always in the "=" column). A few even exhibit counter-bias (marked in *italics* in Table 3; *e.g.*,

script families with pro-stealth bias loading from domains with pro-naive bias in total requests). The overlap between like-biased domain sets (marked in **bold** in Table 3) is significantly larger for intuitive biases (*e.g.*, favoring more realistic clients) than for their counter-intuitive complements, suggesting that intuitive biases reflect intentional, targeted behavior while their counter-intuitive complements contain more random noise. Most importantly, we see measurement bias patterns clearly extending beyond high level request/domain volume statistics and into content itself.

3.3.2 Biases in JS APIs Used. We find examples of behavioral bias at the JS API usage level even within script families loaded with equal frequency across different configurations. A case study of apparent server-side anti-cloud bias is provided by a video player script family served from `facebook.com`. Two variants of this player are consistently loaded during crawls from each VP for 29 sites with some reasonably high Tranco ranks (*e.g.*, #218, `ettoday.net`). The code is not obfuscated and makes references to ad-related features, including "fraud detection," strongly suggesting its purpose is advertising. Analysis of API traces reveals a subtle but universally consistent difference in API footprint between trace from cloud clients and residential/university clients. This cluster of 6 API features is invoked only on a path of execution triggered after a video successfully loads and begins playing. The mismatch implies that on visits from our cloud VP, the video content consistently fails to load and play as it does on the university and residential visits. Location-based content targeting cannot be ruled out as a factor, but it seems odd that apparently *no* videos successfully played on cloud crawls rather than *different* videos, across 29 distinct sites. API traces contain no tell-tale indicators of bot-detection or other environment-sensitive behaviors, implicating server-side logic as the probable culprit for the discrimination.

An unmistakable case of client-side anti-bot discrimination is provided by a family of scripts served by crazyegg.com. We found 4 distinct lexical hashes (representing 161 total distinct scripts) with near-identical code structure and API usage patterns. These scripts were loaded on 158 distinct sites, including highly-ranked domains like `cnet.com`, `nationalgeographic.com`, `umich.edu`. According to VisibleV8 API traces, a typical variant invoked less than 15 browser API features during naive crawls, but nearly 60 API features during stealth crawls. Investigation of the script code and trace data revealed the cause: a function named `uaBot` that performs a battery of increasingly sophisticated and obscure checks for tell-tale artifacts of bots.

Execution traces from naive clients show execution never getting beyond a simple user-agent string test, which fails on the presence of the string "`eadless`". Stealth client traces show execution proceeding through the entire range of tests (due largely to the hardening provided by `puppeteer-stealth`, see Section 2.5) and proceeding to invoke other, more application-oriented API functionality. Note that the majority of API invocations missing from naive traces are not related to the bot tests themselves but are a consequence of the short-circuit failure on bot-detection. Unsurprisingly, `crazyegg.com` scores as a somewhat pro-stealth domain, serving 1.77 times as many requests on stealth crawls than on naive crawls. At least some of this request volume bias is the direct result of a client-side decision made in light of user-agent realism or lack thereof.

3.3.3 Summary. The loading and execution of lexically-distinct families of JS code shows that measurement bias extends beyond HTTP traffic volumes and into the content served. About 10% of script families show significant and consistent bias across BC, accounting for about 15% of all script executions. The vast majority (66% to 89%) of the domains serving this client-biased content showed no significant bias in overall HTTP request volume, leading us to conclude overall measurement bias effects may be even higher than suggested by our simple HTTP request volume bias metrics. Case studies of evasive scripts found in the wild reveal VP and BC biases in runtime behaviors of the same script loaded by different clients.

4 DISCUSSION AND FUTURE WORK

4.1 Application to Future Research

In this work we have found that seemingly minor decisions in web measurement methodology can yield significantly different measurements of privacy and security behaviors. In this subsection we provide some suggestions on how researchers can integrate this work's findings, to improve the likelihood that their measurements will generalize to actual user experiences.

4.1.1 Prefer User-Targeted Tools. While tools built for automated testing are convenient, we recommend researchers take caution before applying them to research purposes. Browser automation frameworks often introduce features that make them look very different from standard browsers, and frequently trigger different code paths than users encounter. Controlling a browser from an extension, or through command line arguments, is less likely to be checked for and detected by page script, and so less likely to result in special casing.

If the research needs capabilities only available through testing-focused tools, as was the case in this work, we recommend using tools[6] that reduce the differences between automated and human operated browsers.

4.1.2 Avoid Headless (or Similar) Modes. Firefox and Chromium-based browsers include a "headless" mode, where pages are rendered and executed, but not displayed to the user. While convenient, we recommend researchers not use such modes. Headless modes have the advantage of enabling more parallelism (because they avoid the overheads needed to render to the user), but also introduce differences in page execution, both directly (*e.g.*, scripts attempting to detect headless mode) and indirectly (*e.g.*, execution optimizations and limits used in headless mode).

Instead, we recommend using stock browsers in their stock configuration wherever possible. While resources are higher, it ensures that the measurement environment is similar to typical user environments. Tools like Xvfb[7] can also help perform headless-like measurements with stock browsers.

4.1.3 Prefer Measurements from Residential IPs. While the parallelism and high-availability of platforms like AWS make them popular for web measurement, we recommend researchers avoid them where possible. Instead we recommend performing user-focused measurements from networks similar to those most web browsing is done from: residential ISPs and mobile carriers. The results in Section 3 demonstrate why we recommend measuring from residential IPs.

[6] *e.g.*, https://github.com/berstend/puppeteer-extra/tree/master/packages/puppeteer-extra-plugin-stealth

[7] https://www.x.org/releases/X11R7.6/doc/man/man1/Xvfb.1.xhtml

We highlight here some of the issues encountered in our residential IP measurements, to help other researchers avoid similar problems. First, we needed to monitor popular IP block lists to make sure the high volume of traffic (compared to typical web browsing) generated by our experiments did not cause our IP to be flagged for suspicious behavior. Second, measuring from a residential IP required reducing the parallelism of our measurements, to avoid adding noise (in the form of bandwidth exhaustion) into our measurements. And third, we took care to consider other network users in ways not needed when measuring from, say, AWS; a couple of Netflix clients are unlikely to impact network behavior at an AWS data-center, but very well could from a residential IP.

4.2 Limitations

Resource constraints restricted our efforts to a single example of each class of VP (cloud, residential, and research university). This set is obviously too small to generalize results across *all* cloud or residential endpoints. Yet the baseline assumption that traffic coming from residential networks is more likely to be end-user traffic (and more likely to be treated as such) than traffic coming from cloud networks is reasonable based on prior experience [12, 24]. The asymmetric nature of the VP bias we observed (*i.e.*, more than twice as many domains showed pro-residential bias as showed pro-cloud bias) minimizes concerns about geographic ad targeting contaminating our results, and strongly implies that cloud VPs are less than ideal for unbiased measurements of end-user concerns.

As noted in Section 2.2, our stealthy BC is not stealthy in an absolute sense (*i.e.*, perfectly approximating human visitors). However, the creation and continued community-maintenance of the stealth plugin we use for our stealthy BC strongly implies that there exists a set of real-world sites that will treat our stealthy BC crawlers as humans while tagging our naive BC crawlers as bots. This difference in outcomes provides us with a lower-bound baseline of measurement bias attributable to bot detection and other related adversarial responses. Our findings therefore likely understate the full potential impact of unrealistic crawlers on measurement results.

4.3 Future Work

Completing this work has caused us to be more sensitive to, and concerned about, other potential sources of bias, blind spots, or threats to generalizability in web measurement research. A partial list of additional possible sources for measurement-bias in web measurement include: (1) differences in "dwell time" on pages during measurements (2) how "child" pages are selected (3) how and whether state is maintained across the crawl (4) operating system selection

In all such cases, the lack of common best practices in measurement invite the possibility that the web measured by researchers may be significantly different than the web experienced by typical users. We plan on future work like this paper, to further understanding and control for sources of measurement bias in web research.

5 RELATED WORK

While much prior work has involved web crawls via various vantage points (VPs) and browser configurations (BCs) and thus intersects ours, we here summarize only a representative sample with special focus on the crawler/human generalization question and other topics involving content blocking or discrimination by the content providers (as opposed to censorship by 3rd parties).

5.1 Generalizability

Zeber *et al.* [26] presented a recent, systematic treatment of how well automated web crawls generalize to web human browsing. Their methodology compared results from multiple automated crawls, varied by VP and client operating system in a manner quite similar to ours, both with each other and with anonymized data collected from 50,000 human volunteers. Our work is scoped more narrowly and deeply: we seek to quantify any gaps in coverage attributable to differential treatment of automated crawls based on VP or BC by performing fine-grained comparisons of HTTP request volume and JavaScript behavior across configurations.

Ahmad *et al.* [7] surveyed web security, privacy, and measurement research literature to quantify both the popularity of different web crawling tools and frameworks and the reproducibility of studies using them. They performed comparative crawls using a representative sample of tools ranging from the primitive (*e.g.*, cURL) to the sophisticated (*e.g.*, OpenWPM) and documented significantly different results tool-by-tool across three example sets of result metrics. Our work is similar in that we compare crawl-to-crawl rather than attempting to quantify crawl-to-human differences. It differs significantly, however, in that we use a much smaller set of realistic tools along with varied network VPs to dig into fine-grained content exclusion practices across these control variables.

5.2 Cloaking and Bot Detection

Invernizzi *et al.* [12] investigated websites attempting to hide their malicious activities. The authors found a large number of websites using IP lists to show benign content to visitors coming from well known measurement IPs, while showing malicious content to other (assumed to be human) traffic. Vadrevu and Perdisci documented examples of social engineering advertisement campaigns that appeared to serve their payloads only to visitors coming from residential networks [24].

Oest *et al.* created PhishFarm [20], a framework for deploying fake phishing sites to evaluate the impact of various cloaking techniques harvested from real-world phishing kits on the crawlers used to maintain various anti-phishing blacklists (*e.g.*, Google SafeBrowsing).

Van Goethem, Le Pochat, and Joosen [25] employed a multi-BC approach to identify dedicated mobile versions of sites served on custom sub-domains and to compare their security practices against their desktop counterparts. While they encountered some instances of malicious cloaking on compromised websites, their focus was on gaining insight into developer security practices over time rather than in deliberate discrimination. Doran and Gokhale [9] surveyed, taxonomized, and compared state of the art web robot detection techniques circa 2010.

5.3 Network Endpoint Discrimination

Much prior work has focused on the design and deployment of systems for detecting when networks and site providers discriminate based on visitor attributes, primarily IP address. Bajpai *et al.* [8] provided a summary of this work, including the strengths, differences, and lineages of existing proposals. In our study, we are concerned

about measuring how the web reacts when visited from different endpoints using both naive and more realistic clients. PacketLab [17] proposed a universal measurement endpoint system by decoupling the measurement logic from the actual system and adopting an access control system for the physical endpoints. In contrast, our architecture is not concerned with endpoint network infrastructure as a packet source/sink but with measuring the impact on web content metrics across multiple endpoints and user agent configurations.

Some researchers have focused on understanding when, why and how websites block IP addresses for security reasons. Khattak *et al.* [15] explored how websites treat requests coming from the Tor network differently than "standard" internet traffic. The work visited the 1k most popular websites and compared how websites respond differently to Tor and non-Tor requests. Afroz *et al.* [6] found that a significant amount of IP-based blacklisting is likely unintended, and the result of overly-general security policies on networks. Tschantz *et al.* [23] looked into a variety of motivations for IP based blocking and found that security was a major motivation, along with political (*i.e.*, GDPR) reasons.

Additional research has explored the motivations for websites presenting different content to users based on their IP addresses. In contrast to cloaking, the content changes here are benign, motivated by regulatory compliance or marketing. Fruchter *et al.* [11] found that websites track users differently, and to varying degrees, based on the regulations of the country the visitor's IP is based in. Iordanou *et al.* [13] described a system for measuring n how e-commerce websites discriminate between users. The authors considered several different motivations for discrimination, including geography (measured by IP address), prior browsing behavior (*e.g.*, tracking-derived PPI) of the user, and site A/B testing. The authors found that the first and third motivations explained more site "discrimination" than the second motivation.

6 CONCLUSION

Where ground truth is drawn from measurement, observations define reality. Our results show that realism in vantage point (VP) and browser configuration (BC) have direct, even dramatic impact on web crawl observations. We found VP bias against our cloud endpoint too asymmetric to be ruled out as geo-targeting. We found BC bias to be especially pronounced in blacklisted ad and tracker traffic, with up to 19% of domains showing significant shifts in traffic volume based on BC. We are hopeful that the contributions we presented in this paper, from the systematic approach to controlled and synchronized web crawls to the guidance offered to those performing future measurements, will be of service to those advancing along this important research front.

Availability: The open source implementation of our experiment is available at https://anonymous.4open.science/r/50f4a903-744f-4de9-adf6-f3dcb44367a3/, along with links to archives of the collected data.

ACKNOWLEDGMENTS

We would like to thank our anonymous reviewers for their insightful feedback and comments. This work was supported by the National Science Foundation (NSF) under grant CNS-1703375 and the NSF Center for Accelerated Real Time Analytics (CARTA).

REFERENCES

[1] [n.d.]. Historical trends in the usage statistics of dns server providers. https://w3techs.com/technologies/history_overview/dns_server. Accessed: 2020-5-29.
[2] [n.d.]. New Industry Benchmarks for Mobile Page Speed - Think With Google. https://www.thinkwithgoogle.com/marketing-resources/data-measurement/mobile-page-speed-new-industry-benchmarks/. Accessed: 2020-5-6.
[3] 2015. GO Simple Tunnel - a simple tunnel written in golang. https://github.com/ginuerzh/gost. Accessed: 2020-06-02.
[4] 2018. . https://antoinevastel.com/bot%20detection/2018/01/17/detect-chrome-headless-v2.html. Accessed: 2020-10-16.
[5] Eytan Adar, Jaime Teevan, Susan T. Dumais, and Jonathan L. Elsas. 2009. The Web Changes Everything: Understanding the Dynamics of Web Content. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining* (Barcelona, Spain) *(WSDM '09)*. Association for Computing Machinery, New York, NY, USA, 282–291. https://doi.org/10.1145/1498759.1498837
[6] Sadia Afroz, Michael Carl Tschantz, Shaarif Sajid, Shoaib Asif Qazi, Mobin Javed, and Vern Paxson. 2018. Exploring server-side blocking of regions. *arXiv preprint arXiv:1805.11606* (2018).
[7] 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 *The Web Conference*.
[8] Vaibhav Bajpai and Jürgen Schönwälder. 2015. A survey on internet performance measurement platforms and related standardization efforts. *IEEE Communications Surveys & Tutorials* 17, 3 (2015), 1313–1341.
[9] Derek Doran and Swapna S Gokhale. 2011. Web Robot Detection Techniques: Overview and Limitations. *Data Mining and Knowledge Discovery* 22, 1-2 (2011), 183–210.
[10] Steven Englehardt and Arvind Narayanan. 2016. Online Tracking: A 1-million-site Measurement and Analysis. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*. ACM. https://doi.org/10.1145/2976749.2978313
[11] Nathaniel Fruchter, Hsin Miao, Scott Stevenson, and Rebecca Balebako. 2015. Variations in tracking in relation to geographic location. *arXiv preprint arXiv:1506.04103* (2015).
[12] Luca Invernizzi, Kurt Thomas, Alexandros Kapravelos, Oxana Comanescu, Jean-Michel Picod, and Elie Bursztein. 2016. Cloak of visibility: Detecting when machines browse a different web. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE.
[13] Costas Iordanou, Claudio Soriente, Michael Sirivianos, and Nikolaos Laoutaris. 2017. 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*. ACM, 376–389.
[14] Jordan Jueckstock and Alexandros Kapravelos. 2019. VisibleV8: In-browser Monitoring of JavaScript in the Wild. /projects/vv8/. In *Proceedings of the ACM Internet Measurement Conference (IMC)*.
[15] Sheharbano Khattak, David Fifield, Sadia Afroz, Mobin Javed, Srikanth Sundaresan, Vern Paxson, Steven J Murdoch, and Damon McCoy. 2016. Do you see what I see? differential treatment of anonymous users. In *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*. Internet Society.
[16] 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 *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*. https://doi.org/10.14722/ndss.2019.23386
[17] Kirill Levchenko, Amogh Dhamdhere, Bradley Huffaker, Kc Claffy, Mark Allman, and Vern Paxson. 2017. Packetlab: a universal measurement endpoint interface. In *Proceedings of the 2017 Internet Measurement Conference*. ACM, 254–260.
[18] J. R. Mayer and J. C. Mitchell. 2012. Third-Party Web Tracking: Policy and Technology. In *Proceedings of the IEEE Symposium on Security and Privacy*.
[19] Fiona Fui-Hoon Nah. 2004. A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & Information Technology* 23, 3 (2004), 153–163.
[20] A. Oest, Y. Safaei, A. Doupé, G. Ahn, B. Wardman, and K. Tyers. 2019. PhishFarm: A Scalable Framework for Measuring the Effectiveness of Evasion Techniques against Browser Phishing Blacklists. In *Proceedings of the IEEE Symposium on Security and Privacy*. 1344–1361.
[21] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. 2012. Detecting and defending against third-party tracking on the web. In *Proceedings of the USENIX symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association.
[22] Peter Snyder, Lara Ansari, Cynthia Taylor, and Chris Kanich. 2016. Browser Feature Usage on the Modern Web. In *Proceedings of the ACM SIGCOMM conference on Internet measurement conference (IMC)*. ACM.
[23] Michael Carl Tschantz, Sadia Afroz, Shaarif Sajid, Shoaib Asif Qazi, Mobin Javed, and Vern Paxson. 2018. A bestiary of blocking: The motivations and modes behind website unavailability. In *8th {USENIX} Workshop on Free and Open Communications on the Internet ({FOCI} 18)*.
[24] Phani Vadrevu and Roberto Perdisci. 2019. What You See is NOT What You Get: Discovering and Tracking Social Engineering Attack Campaigns. In *Proceedings of the ACM Internet Measurement Conference (IMC)*.

[25] Tom Van Goethem, Victor Le Pochat, and Wouter Joosen. 2019. Mobile Friendly or Attacker Friendly? A Large-Scale Security Evaluation of Mobile-First Websites. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security* (Auckland, New Zealand) *(Asia CCS '19)*. Association for Computing Machinery, New York, NY, USA, 206–213. https://doi.org/10.1145/3321705.3329855

[26] 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 *The Web Conference*.