Apophanies or Epiphanies? How Crawlers Impact Our Understanding of the Web

Syed Suleman Ahmad  
University of Wisconsin - Madison

Muhammad Daniyal Dar  
University of Iowa

Muhammad Fareed Zaffar  
LUMS

Narseo Vallina-Rodriguez  
IMDEA Networks / ICSI

Rishab Nithyanand  
University of Iowa

ABSTRACT

Data generated by web crawlers has formed the basis for much of our current understanding of the Internet. However, not all crawlers are created equal and crawlers generally find themselves trading off between computational overhead, developer effort, data accuracy, and completeness. Therefore, the choice of crawler has a critical impact on the data generated and knowledge inferred from it. In this paper, we conduct a systematic study of the trade-offs presented by different crawlers and the impact that these can have on various types of measurement studies. We make the following contributions: First, we conduct a survey of all research published since 2015 in the premier security and Internet measurement venues to identify and verify the repeatability of crawling methodologies deployed for different problem domains and publication venues. Next, we conduct a qualitative evaluation of a subset of all crawling tools identified in our survey. This evaluation allows us to draw conclusions about the suitability of each tool for specific types of data gathering. Finally, we present a methodology and a measurement framework to empirically highlight the differences between crawlers and how the choice of crawler can impact our understanding of the web.

ACM Reference Format:

1 INTRODUCTION

Web crawlers and scrapers have been a core component of the toolkits used by the Internet measurement, security & privacy, information retrieval, and networking communities. In fact, nearly 16% of all publications from the 2015-18 editions of ACM Internet Measurement Conference (IMC) [23], Network and Distributed System Security Symposium (NDSS) [27], ACM Conference on Computer and Communication Security (CCS) [13], USENIX Security Symposium (SEC) [34], IEEE Symposium on Security & Privacy (S&P) [22], International World Wide Web Conference (WWW) [25], International AAAI Conference on Web and Social Media (ICWSM) [24], and Privacy Enhancing Technology Symposium (PETS) [31] relied on Web data obtained through crawlers. However, not all crawlers are created equal or offer the same features, and therefore the choice of crawler is critical. Crawlers generally find themselves trading-off between computational overhead, flexibility, data accuracy & completeness, and developer effort. On one extreme, command-line tools such as Wget [35] and cURL [18] are lightweight and easy to use while providing data that is hardly accurate or complete – e.g., these tools have the inability to handle dynamically generated content or execute JavaScript which restricts the completeness of gathered data [21]. As JavaScript is currently used in over 95% of the Alexa Top-100K web rank (as of August 2019), their use as crawlers in specific types of research is questionable [37]. On the other extreme, tools like Selenium [10] provide the ability to get extremely accurate and complete data at the cost of high developer effort (needed to develop automated web interaction models that can mimic real users) and computational overhead (needed to run a fully-fledged Selenium-driven browser such as Firefox or Chrome). Further complicating matters, in response to a growing variety of attacks, web servers are becoming increasingly sensitive to and willing to block traffic appearing to be anomalous – including traffic appearing to be generated by programmatic crawlers [15, 33].

As a consequence of the above mentioned trade-offs, it is important to understand how the choice of crawler can make an impact on data gathered and inferences drawn by a measurement study. What is lacking, however, is a systematic study of the trade-offs presented by different crawlers and the impact that these can have on various types of measurement studies. In this paper, we fill this gap. Our overall objective is to provide researchers with an understanding of the trade-offs proposed by different crawlers and also to provide data-driven insight into their impact on Internet measurements and the inferences drawn from them. We accomplish this objective by making the following contributions.

Surveying crawlers used in prior work. (§2) We start our study by surveying research published since 2015 at the following venues: IMC [23], NDSS [27], S&P [22], CCS [13], SEC [34], WWW [25], ICWSM [24], and PETS [31]. In our survey of 2,424 publications, we find 350 papers that rely on data gathered by 19 unique crawlers, permitting. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2020 Association for Computing Machinery.
https://doi.org/10.1145/3366423.3380113
on the results of our survey to select crawlers for further analysis. We study the capabilities and flexibility of each of these crawlers with the goal of understanding how they are likely to impact the data collection and their limitations. This allows us to draw conclusions about the suitability of each tool for specific types of research problems.

A framework for performing comparative empirical crawler evaluations. (§4) We present a methodology and accompanying framework for empirically highlighting the differences between crawlers in terms of the network and application data they generate. Further, in an effort to guide future researchers in making a choice between different crawlers in specific scenarios, and to make our work repeatable, we make this evaluation framework available publicly at https://sparta.cs.uiowa.edu/projects/methodologies.html.

Illustrating the impact of different crawlers on research results and inferences. (§5) We empirically demonstrate how different crawlers impact measurement results and inferences drawn from them (and consequently: our understanding of the web). Based on large amounts of prior work, we focus on the three major problem domains: privacy, content availability, and security measurements. Through our experiments, we observe the impact on the inferences made on the (1) success of traffic analysis classifiers, (2) third-party ecosystem, and (3) incidence rates of server-side blocking. This illustrates that crawler selection does indeed impact measurement and experiment quality, soundness, and completeness.

2 SURVEY OF CRAWLING METHODOLOGIES

We begin our study by surveying the use of crawlers in recent academic research conducted by the Internet measurement, security, and privacy communities. Our goal is to understand how these research communities use web crawlers and to verify the repeatability of these crawls.

2.1 Survey methodology

To achieve our goal of understanding how crawlers were used by researchers in the fields of Internet measurement, security, and privacy, we conducted a survey of the publications from the 2015-2018 editions of the following conferences: ACM Internet Measurement Conference (IMC) [23], Network and Distributed System Security Symposium (NDSS) [27], ACM Conference on Computer and Communication Security (CCS) [13], USENIX Security Symposium (SEC) [34], IEEE Symposium on Security & Privacy (S&P) [22], International World Wide Web Conference (WWW) [25], International AAAI Conference on Web and Social Media (ICWSM) [24], and Privacy Enhancing Technology Symposium (PETS) [31]. At a high-level, the survey methodology is designed to: (1) identify the crawling method and tool used in each publication — these results are used to inform our analysis in §3; (2) verify repeatability — i.e., whether the crawler is used in a repeatable manner; and (3) identify the purpose and research domain of the experiment and the associated publication — these results are used to inform our evaluation in §5.

Crawler identification. Our first goal in this survey is to identify which crawlers were most commonly used by the research community. To accomplish this goal, we used the following filtering process. First, we randomly sampled 300 publications for manual inspection from the total set of 2,424 papers. During this manual inspection process, we extracted keywords that were observed to be commonly associated with crawling (Table 1). Next, we filtered out publications not containing these keywords based on the idea that these were unlikely to include a crawler as part of their data gathering. This left us with a total of 764 papers. Each of these was then manually inspected to verify the use of a crawler for data gathering. We found a total of 350 publications meeting this criterion. This manual inspection helped us to eliminate false-positives generated by our keyword-based heuristic, often due to mentions of crawling tools and browsers in the related work sections of these papers. During this second round of manual inspection, we identified the use of 19 unique crawling tools in 350 papers. We labeled papers as having used custom-built crawlers if the crawling tool was developed for the purpose of that specific study. For example, several studies leveraged custom-built extensions or app-store specific crawlers to overcome the inadequacies of popular off-the-shelf crawlers [38, 43]. Although we include statistics on the occurrences of custom tools, we do not consider them as part of our study. Additionally, we do not include papers relying on data generated using RESTful APIs as part of our study since RESTful API responses are based on a protocol and are not impacted by crawler or browser technologies.

Repeatability verification. In order for a data-gathering crawl to be repeatable, we expect that the associated scientific publication at least specifies and describes the crawling tool used for data gathering. We argue that in the absence of this information, it is impossible for other researchers to accurately replicate the data-gathering mechanism. Therefore, while conducting our manual inspection of the 350 papers using crawlers for data collection, we label those that do not report the tool or framework used as not repeatable. We note that this is in fact a lower-bound on non-repeatability since it is also important to have crawler-specific configuration details. For example, a study that simply mentions the use of the cURL crawler is marked as repeatable in our survey when in reality additional configuration details such as whether the crawler was configured to follow redirects, etc. are also required to accurately replicate the study. We note that studies leveraging custom crawlers were labeled as repeatable to satisfy our lower-bound. This is reasonable because papers using custom tools often explicitly specify their implementation details. We identified 136 studies which were not repeatable based on this definition.

<table>
<thead>
<tr>
<th>Category</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>crawler*, scrap*, webkit, webdriver, spider, browser, headless</td>
</tr>
<tr>
<td>Browsers</td>
<td>Firefox, Chrome, Chromium</td>
</tr>
<tr>
<td>Tools</td>
<td>cURL, Wget, PhantomJS, Selenium, OpenWPM</td>
</tr>
</tbody>
</table>

Table 1: Keywords obtained from a manual analysis of 300 publications.
Research topic categorization. In order to make inferences about the types of crawling tools and methodologies preferred by researchers in different problem domains and scenarios, we taxonomized the research domains in which crawlers were frequently used. While manually inspecting the 350 publications, we classified each as belonging to one or more of the following eight classes: privacy, censorship, performance measurements, content testing, security measurements, web mining, automation, and surveying. These topics are described in Table 2.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privacy</td>
<td>Crawler-gathered data was used for the measurement and analysis of online advertising and tracking ecosystems.</td>
</tr>
<tr>
<td>Censorship</td>
<td>Crawler-gathered data was used for the measurement and analysis of Internet censorship and circumvention.</td>
</tr>
<tr>
<td>Performance</td>
<td>Crawler-gathered data was used to evaluate the performance (in terms of page-load times and bandwidth) of different networks, architectures, and tools.</td>
</tr>
<tr>
<td>Content</td>
<td>Content-gathered data was used to analyze the structure of webpages (DOM and element presence) and to fuzz test web applications.</td>
</tr>
<tr>
<td>Testing</td>
<td>Content-gathered data was used to identify and characterize the use of different security protocols and mechanisms.</td>
</tr>
<tr>
<td>Web mining</td>
<td>Crawler-gathered data was used to understand web service usage and publication mechanics – e.g., scraping the Google Play store to gather data about the release schedule of Android app updates.</td>
</tr>
<tr>
<td>Automation</td>
<td>Crawlers were used to automate interactions with online services – e.g., automating account creation.</td>
</tr>
<tr>
<td>Surveying</td>
<td>Crawlers were used as a component of experiments and studies aimed at understanding human subjects – e.g., tracking eye movements to understand human perception of page-load completion.</td>
</tr>
</tbody>
</table>

Table 2: Topics used to categorize the final list of 350 publications which relied on crawlers for data gathering. A publication was placed in a category based on the purpose for which crawler-gathered data was used. Note that a publication may be assigned multiple classes.

2.2 Results

Which crawlers are most popular? In our manual inspection of 350 publications, we identified a total of 126 publications (36%) relying on one of the 19 different crawlers shown in Table 3. The most commonly observed subset of the 19 crawlers are broken down by their use in different years, conference venues, and research domains in Table 4. Generally, we observe that there are a small number of popular crawling tools (4: Selenium, PhantomJS, cURL, and Wget) which were used by over one-quarter of all research papers relying on crawlers for data collection and analysis. In the remainder of this paper, we use these crawlers as the basis for comparison and evaluation.

What fraction of crawls are not repeatable? We now turn our focus to understanding the characteristics of crawls that we labeled as not repeatable. As described in §2.1, a crawl was labeled “not repeatable” if the corresponding research paper did not mention the crawling tool or framework used to generate the crawled data. While this metric is a strict lower-bound for the actual non-repeatability of crawls, we found that 36% of research papers analyzed in our survey meet this definition. Interestingly, we note that the incidence of non-repeatable crawls is most evident in publications at WWW suggesting the need for additional research exposition and review considerations.

What research domains are most reliant on crawlers? Based on the methodology outlined in §2.1, we find that crawlers were most commonly used for the purposes of web mining, content testing, automated web interaction, and privacy measurements. A breakdown across all areas is shown in Table 4. Interestingly, we find that custom crawling solutions are most commonly observed in the content testing and security categories. This suggests that existing tools need to be augmented to fit the needs of each research domain. We also note that non-repeatable crawls were most common for web mining, performance testing, and automated web interaction purposes. This supports our previous suggestion of the need for additional research exposition and review considerations.

In our analysis of the impact of different crawling methodologies on research inferences (§5), we focus on research in the privacy, content testing, and security communities since they generally generate repeatable crawls and simultaneously show high variation in the tools that they rely on for these crawls.

Takeaways. We see that web crawls form a major component of the Internet measurement, security, and privacy communities with over 16% of all research papers in the last four editions of the premier venues relying on data from crawlers. Rather worryingly, our survey also highlights the lack of specificity in research publications when describing crawling methodologies. In fact, over 35% of the crawl-dependent publications were not repeatable due to absence of critical information regarding crawl methodology. Finally, we observe that crawl repeatability, incidence of custom solutions, and variation across crawling tools are all dependent on the research domain. We use the most popular crawling tools uncovered from this survey as the basis for our analysis in the remainder of this paper and the research domains showing most variance across crawling tools as the basis for our analysis in §5.

3 QUALITATIVE EVALUATION

In this section, we conduct a qualitative comparative evaluation of a selected set of eight crawling tools uncovered by our survey (§2). The chosen crawlers (listed in Table 5) reflect the most popular ‘out-of-the-box’ solutions that are used by the research studies analyzed in our survey. Our primary focus is on understanding the capabilities and flexibility afforded by each of these tools with an eye on how these might impact the data generated by them.
Crawling tools broken down by year and publication venue. Note that a publication could use multiple crawling tools.

<table>
<thead>
<tr>
<th>Year</th>
<th>Total</th>
<th>Publication venue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015</td>
<td>523</td>
<td>S&amp;P IMC PETS SEC NDSS CCS WWW ICWSM</td>
</tr>
<tr>
<td>2016</td>
<td>117</td>
<td>S&amp;P IMC PETS SEC NDSS CCS WWW ICWSM</td>
</tr>
<tr>
<td>2017</td>
<td>335</td>
<td>S&amp;P IMC PETS SEC NDSS CCS WWW ICWSM</td>
</tr>
<tr>
<td>2018</td>
<td>335</td>
<td>S&amp;P IMC PETS SEC NDSS CCS WWW ICWSM</td>
</tr>
</tbody>
</table>

3.1 Evaluation methodology
Characterizing crawling approaches. We begin our evaluation by first categorizing crawlers by their layer of operation. We make this categorization based on the observation that at a fundamental level, web crawlers are tools to generate application-layer protocol requests and process corresponding responses. The generation and processing of these requests and responses may occur in different ways. For example, some crawlers may directly interface with the application-layer of the networking stack to create (e.g., HTTP GET) requests with specific parameters, some may rely on controlling a browser to generate application-layer requests, and others may rely on mimicking user interactions with a browser to generate these requests. More concretely, we classify crawling tools into one of the following three categories.

Application-layer protocol crawlers (ALPCs). Crawlers in this category operate by handling web content as static files and perform HTTP GETs or POSTs to “crawl” a website. They do not interface with web browsers. cURL, Wget, and ZAP spider fall in this category.

Browser-layer crawlers (BLCs). Crawlers in this category operate by driving web browsers. These crawlers typically provide wrappers in many popular languages to interface with browser-provided JavaScript APIs which interact and manipulate the DOM. For example, crawler-provided functions to navigate to specific webpages work by generating JavaScript which manipulates the window.location.assign property in the DOM of Blink-based browsers. PhantomJS, Selenium, and Puppeteer fall in this category.

User-layer crawlers (ULCs). Crawlers in this category operate by implementing a wrapper around standard browser-layer crawlers. These wrappers introduce new functionalities, such as mimicking user interaction with loaded pages, by chaining together multiple methods provided by browser-layer crawler bindings, therefore reducing the manual effort required to conduct specific types of crawls. OpenWPM and TorBC fall in this category.

Characterizing features of crawling tools. We characterize the features of different tools by the flexibility they offer to: (1) manipulate application-layer protocols (e.g., HTTP request parameters), (2) modify browser or client characteristics (e.g., enabling cookies, caching, or extensions), (3) automate user-interactions (e.g., enabling rate-limiting, automatic anti-bot detection mechanisms, etc.), (4) facilitate data gathering (e.g., saving screenshots, network traces, etc.), and (5) automate and scale-up crawls (e.g., permitting headless crawls, parallel crawling, etc.). Each of these categories is described in more detail below.

Application-layer protocol-level features. Features in this category capture the ability of a crawler to manipulate the parameters associated with typical web protocols. The ability to modify these protocol characteristics is particularly important for measuring the support of security protocols and characterizing server-side protocol behaviors – e.g., measuring server behaviour in response to different TLS options and HTTP headers. In our evaluation, we focus on the ability of the crawler to add new or modify existing HTTP headers, toggle the HTTP keepalive option, create new or manipulate generated HTTP methods (e.g., GET and POST), and set SSL/TLS options (e.g., version number).

Browser(client)-level features. Features in this category generally capture the ability of a crawler to manipulate browser characteristics. Browser characteristics have an impact in the types of requests generated by the crawler – e.g., clients configured to disable JavaScript will generate fewer requests for third-party content [49]. Therefore, it is crucial to be able to manipulate them. The features
that we consider in this study are related to: (1) determining how the client handles web content – e.g., can the tool be set/configured to ignore locally cached content, not execute JavaScript, disable cookie access; (2) browser-specific settings including which extensions should be loaded and whether proxies can be specified.

User-level features. Traffic generated by crawlers are increasingly seen as unwanted and often characterized as attack traffic by web administrators [15, 33]. Consequently, it is becoming increasingly important for researchers conducting web measurements to take steps to ensure that their crawls do not trigger events that result in blocking or differential treatment. This is especially important for studies seeking to attribute failed page loads to server-side policies [65, 76] or censorship [72, 80]. This requires some crawl characteristic manipulation in order to match crawls to typical human behavior (or at the very least: not look bot-like). Therefore, we evaluate each crawler on its ability to evade server-side blocking by emulating human characteristics. Specifically, we ask the following three questions of each crawler: Does the crawler provide a mechanism to (1) restrict the rate of crawling, (2) dynamically interact with a webpage – e.g., through clicking and typing, and (3) trigger mouse movements and rate-limiting keystrokes?

Data gathering features. Features in this category capture the variety of data that a crawler might facilitate. We consider whether a crawling tool provides access to data at the network/transport layer (packet captures), application layer (HTTP requests and responses), and content layer (screenshots and content fetches). Access to each of these is critical for different problem domains – e.g., packet captures are crucial for understanding the fingerprintability of webpages [42, 79], request and response pairs are crucial for understanding how trackers and third-party content are loaded [47, 49], and screenshots of pages can be crucial for automating the process of blockpage and censorship identification [76].

Crawl-level features. Generally speaking, conducting large-scale web crawls can be an expensive task to undertake – computationally and manually. Therefore, we consider whether each crawler provides features to ease the computational and developer efforts required to use them. Specifically, we check to see if they are capable of automatically splitting workload across multiple parallel crawls (multi-instance), able to conduct crawls in headless mode – i.e., a browser with in-memory DOM rendering, and able to conduct crawls using full-fledged browsers such as Chromium and Firefox.

3.2 Results
A complete breakdown of our evaluation is shown in Table 5. Below, we show how the characteristics of crawling tools and their features result impact their suitability for different tasks.

Rich crawl-level and data gathering features come at the cost of protocol-level flexibility. As one might expect, ULCs generally provide the most number of crawl-level and data gathering features. This is unsurprising as ULCs are explicitly designed with the goal of simplifying data collection and facilitating parallelized BLCs. Unfortunately, this comes at a cost – the ability to manipulate lower-level details such as application-layer protocol parameters. Conversely, while ALPCs provide direct access to these parameters, the data they facilitate is incomplete due to the inability to actually render requested content or load dynamic content (since they treat web content as files and do not present a rendering or execution environment). Some BLCs offer a happy medium between these extremes – providing the flexibility to enable nearly all these features through customized scripts. This suggests that ALPCs are more suitable for crawls which seek to uncover server behavior in response to different headers (e.g., for security fuzzing, network performance measurements, etc.) while studies which seek to measure user experiences (e.g., for censorship, privacy measurements, page load performance, etc.) need to rely on BLCs or ULCs.

BLCs provide the richest user-level and browser-level features. BLCs which drive standard browsers offer the capability to manipulate all the user- and client-level features we considered. This is not possible with ALPCs since they do not use browser environments. Rather surprisingly, we found that the implementations of the ULCs in our study often removed access to important user- and browser-level features that are available in their underlying BLC (e.g., the proxy and caching options for both and all user-level features for TorBC). This suggests that studies which need to avoid bot detection (e.g., content testing required in censorship and privacy measurement) and manipulating local state (e.g., network performance measurements) are more suitable for BLCs and ULCs.

Takeaways. Our analysis hints that specific crawling tools are more suitable for use in different research domains and that the incorrect choice of crawler may significantly impact research results. For example, using an ALPC to gather network traces for input to a website fingerprinting classifier model might over-estimate the attack’s success rate due to the lack of dynamic content. Similarly, studies seeking to understand censorship or differential treatment of users will likely have different results when measured using BLCs and ULCs which incorporate bot-detection mitigation. We empirically evaluate several similar hypotheses in §5.

4 A COMPARATIVE FRAMEWORK
In this section, we describe our framework for empirically highlighting the differences in the network traffic generated between the crawlers described in §3. Table 6 shows the version information of crawlers used in this framework. Our evaluation framework focuses on uncovering the impact of crawlers on requests generated and responses received by clients and traffic characteristics observed by the network. Our hope is that this framework will enable more complete, sound, and correct research which considers and reports the impact of different crawling methodologies on their hypotheses.

4.1 Design considerations
Our framework was designed with two specific goals: extensibility and correctness. To achieve high extensibility, we seek to make our framework modular and easily configurable. Achieving correctness is more challenging due to a number of confounding factors when performing comparative crawls, including: (1) websites are frequently updated and crawls starting at different times may observe different content, (2) web servers may provide localized results making it challenging to rely on cloud infrastructure for measurements, and (3) websites might trigger IP address based discrimination against IPs suspected to be associated with crawlers.
The current implementation of the master only requires three configuration parameters: a list of worker machines, available IP addresses from which the crawls are to be conducted, and target domains to crawl. Each worker machine only needs to implement the command line instruction to launch the local wrapper program. Workers only require two configuration parameters: the master’s IP address and the command line instruction to launch the local wrapper program.

Perform crawl synchronization. We make an effort to achieve a reasonable level of synchronization between crawls from different tools. Our master script coordinates efforts in such a way that all crawls this domain using any crawling tool. Our first release includes wrappers for the eight different crawlers. Workers only require two configuration parameters: the master’s IP address and the command line instruction to launch the local wrapper program.

To achieve extensibility and address the above confounding factors which impact correctness, we take the following steps:

**Rely on an easily configurable master-worker architecture.**

The current implementation of the master only requires three configuration parameters: a list of worker machines, available IP addresses from which the crawls are to be conducted, and target domains to crawl. Each worker machine only needs to implement a wrapper program which takes as input a domain name. It then crawls this domain using any crawling tool. Our first release includes wrappers for the eight different crawlers. Workers only require two configuration parameters: the master’s IP address and the command line instruction to launch the local wrapper program.

**Perform crawl synchronization.** We make an effort to achieve a reasonable level of synchronization between crawls from different tools. Our master script coordinates efforts in such a way that all workers begin crawling a specific webpage at approximately the same time (granularity of seconds). This allows us to negate the impact of frequent website updates within a small margin of error.

**Account for changing IP reputations.** To account for web servers discriminating our workers based on blacklisting and IP reputation scores, we integrate Cisco’s public Talos Intelligence in our master program. Talos returns one of three classes for each queried IP address – ‘Good’, ‘Neutral’, or ‘Poor’. If any worker is found to have an IP reputation that is lower than the others, a new IP address with similar reputation to the IPs used by other workers is dynamically reassigned to it. Therefore, each worker is conducting its crawl from an IP address of comparable reputation to the other worker IP addresses. A limitation of our approach is that we rely on intelligence data from a single source – Talos [16]. However, our IP intelligence class can be easily extended to use APIs from other intelligence services.

### 4.2 Deployment for this study

**Master and worker machine setup.** We used one virtual machine for each worker and master – each with a unique IP address. In total, we had eight workers – one for each crawler. Each worker was configured to use its crawling tool in its default configuration, with the exception of OpenWPM for which we also enabled its “anti-bot detection” functions.

**IP addresses.** We obtained 22 unique IPv4 addresses which were used in our crawl. These IPv4 addresses were obtained from a regional ISP for the duration of these experiments and were found to have a “Good” IP reputation at the start of our measurements. Our IP addresses were geolocated to the same region thereby reducing the impact of content localization.

**URL list.** Our URLs included the union of the Top 500 domains from Alexa [14] and Umbrella’s [17] top sites lists. In total, there were 932 unique domains (of which 30 domains were removed as they returned SOA records) which were crawled by each of our workers. The domain lists were fetched on 25th April 2019.

### 5 Quantitative Evaluation

In this section, we empirically demonstrate how different crawling tools impact measurement results and their resulting inferences. The analysis in this section is based on the data generated by our comparative framework (§4) using the parameters described in §4.2.
5.1 Generic data differences

We start by analyzing the differences in data generated by each of our studied crawlers. We analyze these differences in two distinct vantage points: the differences observed by end hosts (e.g., requests generated by clients and responses received from servers) and the differences observed by the network (e.g., flow characteristics).

**Host-observed differences.** Our analysis shows that different crawler classes (ALPC, BLC, and ULC) cause different characteristics in requests and responses. We highlight two of these below.

ALPC request and response characteristics are drastically different from BLCs and ULCs. As one might expect, the inability of ALPCs to load dynamic content makes the characteristics of their requests and responses very different from those generated by BLCs and ULCs. Our analysis that the three ALPCs (`wget`, `curl`, and ZAP) in our study have several orders of magnitude fewer destinations that they send requests to, in comparison to BLCs and ULCs, while loading a webpage. Further, the sizes of these requests (shown in Figure 1a) and the corresponding responses (shown in Figure 1b) are also orders of magnitude smaller. In comparison, we see smaller variations between different ULCs and BLCs. We attribute these variations to the differences in the browsers (and the browser versions) that they drive – suggesting that crawler specifications are important for accurate reproduction of research and measurement findings. These findings further suggest that ALPCs, despite their favorable performance, are not suitable for content analysis or third-party ecosystem measurements.

![Figure 1: Host-observed differences between crawlers.](image)

Ciphersuite measurements are crawler dependent. We find that there are not only notable differences in the ciphersuites made available by each crawler during TLS negotiation, but also noticeable differences in the negotiated ones. When simply considering the availability of ciphersuites, we see that ALPCs once again stand out (offering over 50 different ciphersuites in TLS negotiation) in comparison to ULCs and BLCs which typically offer less than 20. It is important to note that while the suites offered by ULCs and BLCs are simply those offered by the browser driven by them, there are still significant differences since not all crawlers drive the same browser versions. For example, at the time of our experiments OpenWPM was unable to drive Firefox versions later than v52.9, while Selenium drove Firefox v66.0. Both browser releases were over a year apart and support quite different ciphersuites. The fraction of overlapping ciphersuites offered by the latest releases of our crawlers is shown in Figure 2. As expected, this difference in ciphersuite support has an impact on the finally negotiated TLS ciphersuite with servers. Measuring the usage of different ciphersuites has been utilized in prior studies to uncover the usage of broken cryptographic schemes [63], understand energy consumption [67], and identify how censorship circumvention traffic may better blend in with uncensored traffic [53]. Our findings suggest that such research needs to account for the fact that their conclusions may differ based on the choice of crawler (and consequently the browser/version) used in their measurements.

![Figure 2: Ciphersuite Overlap amongst crawlers.](image)

**Network-observed differences.** In addition to differences at the end-hosts, we find that the choice of crawler also has an impact on the traffic flow characteristics observed by the network. Specifically, we see that: burst characteristics vary by crawler. At the network-layer a burst is characterized by a sequence of consecutive packets (or bytes) all flowing in the same direction. Our analysis shows that the size of bursts is crawler dependent. Specifically, we find that ALPCs show much lower variations in burst sizes while more complex crawlers show much higher variation. TorBC is an exception since it relies on the Tor Browser which uses padding to prevent traffic analysis. Our finding makes intuitive sense: crawlers operating complete browsers are likely to have more bursts and variations in their sizes due to dynamic content on webpages. Further, we also see small variations in the mean burst sizes across different BLCs. This finding is particularly relevant to traffic analysis and TLS fingerprinting studies seeking to identify different classes of traffic based on network flows [51, 55, 74] since it shows that classifying traffic generated by different crawlers may yield different results.

Crawl performance characteristics vary by crawler. The rate of page requests is an important aspect to consider when selecting...
a crawler. As one might expect, the crawling rate (pages loaded per hour) is highest for ALPCs and lowest for ULCs. The differences between crawler page-load times is shown in Figure 3. Each point represents a successful page-load with a response status of 200 and its corresponding x- and y-axis values denote the time-to-load and bytes transferred. We see that ALPCs occupy the bottom left of the plot and ULCs occupy the top right corner. This plot, although difficult to inspect, hints at the inherent trade-offs at play when selecting a crawler: ALPCs provide high crawling performance at the cost of completeness while ULCs provide completeness at the cost of performance.

![Figure 3: Distribution for crawling time vs content size.](image)

### 5.2 Impact on research inferences

Our previous results (§5.1) have shown that crawling method can have a significant impact on lower-level measurements pertaining to characteristics of client-generated requests, server-generated responses, and network flows. In this section, we seek to understand how these differences can impact higher-level measurements which seek to characterize the state of the web. Specifically, we use data generated from our crawling framework to test the following hypotheses: \( H_{wf} \) Choice of crawling method impacts the fingerprintability of websites, \( H_{tp} \) choice of crawling method impacts the measurements of prevalence and prominence of online tracking third-parties, and \( H_{sb} \) choice of crawling method impacts measurements seeking to uncover server-side blocking (e.g., for censorship or discrimination measurements).

<table>
<thead>
<tr>
<th>Method</th>
<th>NB</th>
<th>MB</th>
<th>JS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wget</td>
<td>95%</td>
<td>94%</td>
<td>97%</td>
</tr>
<tr>
<td>cURL</td>
<td>96%</td>
<td>92%</td>
<td>97%</td>
</tr>
<tr>
<td>OWASP ZAP</td>
<td>90%</td>
<td>85%</td>
<td>95%</td>
</tr>
<tr>
<td>PhantomJS</td>
<td>92%</td>
<td>89%</td>
<td>95%</td>
</tr>
<tr>
<td>Selenium</td>
<td>86%</td>
<td>72%</td>
<td>91%</td>
</tr>
<tr>
<td>Puppeteer</td>
<td>90%</td>
<td>89%</td>
<td>93%</td>
</tr>
<tr>
<td>OpenWPM</td>
<td>92%</td>
<td>90%</td>
<td>96%</td>
</tr>
<tr>
<td>TOR BC</td>
<td>2%</td>
<td>5%</td>
<td>4%</td>
</tr>
</tbody>
</table>

**Table 7:** WF accuracy on traces generated by different crawlers.

**Hypothesis \( H_{wf} \):** Choice of crawling method impacts the fingerprintability of websites. In order to validate this hypothesis, we need to show that the success rates of website fingerprinting classifiers are impacted by the crawling method used to generate the data input to the classifiers.

**Methodology.** We selected three simple classifiers from previous website fingerprinting research in our study: a Naive Bayes (NB) classifier [64], a Multinomial Bayes (MB) classifier [57], and a Jaccard Similarity (JS) classifier [71]. We note that although these classifiers are not the current state-of-the-art, they are sufficient to prove our hypothesis. Implementations of these classifiers were obtained from [58, 78]. Data was generated by our crawling framework. The URL list (described in §4) was crawled through each crawler 10 times and traces for each domain and each iteration were logged. These traces were split into training (70%) and testing (30%) datasets and used as input to our classifiers.

**Results.** Our results are highlighted in Table 7. Here we see that TorBC has the lowest fingerprintability since it has explicit defenses to prevent such attacks. However, amongst other crawlers we see that there is between a 2-22% difference in accuracy for the same fingerprinting classifiers and only changing the crawling tool. While this is sufficient to prove our hypothesis, we can show even more significant implications of not using uniform crawler configurations when conducting evaluations of classifier effectiveness. Consider a comparison of the JS and NB classifier both evaluated using OpenWPM – we see that the JS classifier has a success rate of 96% and the NB classifier has a success rate of 92%. This leads us to conclude that the JS classifier is more effective at launching website fingerprinting attacks on the specific list of domains. However, by evaluating the JS classifier using Selenium or the NB classifier with any ALPC we might conclude the opposite. It is important to note that while our toy experiment does not go into comprehensive evaluations of the state-of-the-art, the implications of not specifying the crawling methodology or not maintaining consistent methodology in traffic analysis research are still significant and valid.

**Hypothesis \( H_{tp} \):** Choice of crawling method impacts the measurements of prevalence and prominence of online tracking third-parties. In order to validate this hypothesis, we need to show that the prevalence and prominence rankings of embedded third-party advertising and tracking domains vary by crawler.

**Methodology.** In order to identify third-party advertising and tracking related domains we rely on EasyList and EasyPrivacy [20]. While these lists may be incomplete, they must only show that results per crawler change. We execute the rules in these lists to identify counts of requests to third-party advertising and tracking domains. We then rank these domains by two metrics for each crawler: prevalence and prominence. We borrow metrics for prominence and prevalence from Englehardt et al. [49]. The general idea is that rankings of prevalence capture the exposure of different trackers to the web – since it is assumed that all sites are equally likely to be visited by users. On the other hand, rankings of prominence capture the exposure of different trackers to actual users – since it is assumed that users are likely to visit popular domains more than less popular domains (i.e., the power law distribution applies to website popularity). Finally, we compare the differences in the prevalence and prominence rankings obtained by data generated from each crawler. We note that our results are impacted by our inability to measure the inclusion chains of third-party resources for all crawlers – a prohibitively expensive proposition. This might introduce variability across crawls due to real-time bidding (RTB)
Table 8: Types of block pages observed by different crawlers. **Success** denotes the fraction of pages that were successfully loaded, **4XX** denotes the rate of 4XX status code errors, **GEO** denotes the rate of regional blocking, **BRO** is the rate of browser errors, **CAP** is the rate of CAPTCHA errors, and **ABU** denotes the rate of IP-abuse based blockpages. Note that ZAP Spider does not capture HTML files, hence we fail to capture blockpage HTMLs.

| Methodology: In order to identify server-side blocking, we rely on methodologies constructed from prior work [60, 68]. Based on previous repositories of block pages [1, 2, 68] and manually identified block pages observed by our own crawls, we crafted generalized regular expressions to identify occurrences of the same or other block pages. Our regular expressions were crafted to identify blocking and discrimination due to HTTP 4XX status errors (i.e., client-side errors typically returned for unauthorized or unexpected client interactions with servers), presence of CAPTCHA errors (typically returned when bot activity is suspected), outdated browser errors (returned when websites cannot be loaded or rendered correctly due to browser characteristics), geo-blocking (i.e., when websites block traffic originating at different regions) [66], and IP abuse (i.e., when the server suspects that the client is behaving maliciously). It is important to note that it is common for servers to simply return HTTP 4XX errors rather than more specific blockpages.

**Results.** Our results are shown in Table 8. Here we see obvious differences in the way servers respond to requests generated by different crawlers – i.e., we have a variation of over 16% in the number of successful page loads. This immediately validates our hypothesis that different crawling tools can significantly impact the inferences drawn from content accessibility measurements. In fact, simply changing our crawling tool, we are able to uncover over 160 sites which actually do implement some form of server-side blocking. Further inspecting the server-provided reasons for blocking, we see that a small fraction of servers consistently provide detailed reasons for blocking – regardless of crawler choice, while a majority simply return an HTTP 4XX status error. Rather surprisingly, we see that ALPCs are not blocked as frequently as previously expected, *with the lowest numbers of both 4XX-Errors and CAPTCHA-based blocking*. We hypothesize that this is because popular bot detection tools require JS execution (e.g., Distil Networks [15]). This suggests that they are suitable for measurements of content reachability, which do not require dynamic loading or user interactions. We also see OpenWPMs bot-mitigation method results in the *least instances of CAPTCHA-based blocking among the BLCs and ULCs*, thus highlighting server-side blocking against bot-like behavior and also the potential use of OpenWPM as a viable circumvention tool. Note

Hypothesis $H_{sb}$: Choice of crawling method impacts measurements of server-side blocking. In order to validate this hypothesis, we need to show that different crawling methods incur different levels of server-side blocking.

Figure 4: Prevalence and prominence rankings of third-party advertising and tracking domains identified by each crawler. The OpenWPM crawler is used as the baseline ranking and darker and lighter colors show how much or how little results from other crawlers deviated from this baseline.

<table>
<thead>
<tr>
<th>Methodology:</th>
<th>OWASP ZAP</th>
<th>cURL</th>
<th>PhantomJS</th>
<th>Selenium</th>
<th>Puppeteer</th>
<th>OpenWPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
<td>82.94%</td>
<td>82.6%</td>
<td>81.97%</td>
<td>78.22%</td>
<td>77.04%</td>
<td>93.13%</td>
</tr>
<tr>
<td>4XX</td>
<td>5.36%</td>
<td>4.61%</td>
<td>6.00%</td>
<td>7.40%</td>
<td>8.69%</td>
<td>7.51%</td>
</tr>
<tr>
<td>GEO</td>
<td>2.68%</td>
<td>2.68%</td>
<td>0.54%</td>
<td>2.68%</td>
<td>2.79%</td>
<td>2.68%</td>
</tr>
<tr>
<td>BRO</td>
<td>0.43%</td>
<td>0.43%</td>
<td>0.54%</td>
<td>0.54%</td>
<td>0.54%</td>
<td>0.54%</td>
</tr>
<tr>
<td>CAP</td>
<td>9.01%</td>
<td>9.09%</td>
<td>11.48%</td>
<td>11.59%</td>
<td>11.16%</td>
<td>10.94%</td>
</tr>
<tr>
<td>ABU</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

**Success** denotes the fraction of pages that were successfully loaded, **4XX** denotes the rate of 4XX status code errors, **GEO** denotes the rate of regional blocking, **BRO** is the rate of browser errors, **CAP** is the rate of CAPTCHA errors, and **ABU** denotes the rate of IP-abuse based blockpages. Note that ZAP Spider does not capture HTML files, hence we fail to capture blockpage HTMLs.
that we do not see instances of IP-Abuse based blockpages, indicating that the IP-reputation checks and dynamic realignment in §4 is successful in removing IP-reputation as a confounding variable. 

**Takeaways.** Our quantitative analysis shows that the data generated by different crawlers clearly impacts lower-level measurements (§5.1) and that crawlers can also impact inferences from higher-level measurement (e.g., $H_{wf}$, $H_T$ and $H_b$). These findings highlight that researchers must not only consider crawler categories, but must also consider the underlying browser-engine used by the crawler, as that have a significant influence on the results generated.

6 RELATED WORK

As shown in our literature review (§2), the use of web crawlers in academic research is ubiquitous, and there have been numerous works seeking to improve our understanding of the implications of different crawling approaches. While Yadav and Goyal [81] focus on the comparison of 12 open-source crawlers, following the same direction as our work for determining suitability of each tool, more specifically and, similar to Ricardo and Serrao [73], their work is limited to comparative analysis of crawlers in terms of scalability and performance. Avraam et al. [40] review and compare focused-web-crawling techniques by examining the proceedings of multiple web conferences. Glez-Peña et al. [54] assess existing commercial scraping frameworks by identifying their strengths and limitations in terms of data extraction capabilities, under the umbrella of biomedical applications. Kumar et al. [61] carry out a large literature survey to identify and compare types of crawling strategies, techniques and policies for information retrieval through search engine crawlers. Likewise, Anu et al. [39] survey and compare various crawling techniques specifically for mining web forums. Becchetti et al. [41] studied and compared the bias induced by different methods of web sampling. While all these studies carry out various comparisons of crawling methods at some level, our contributions differ by providing both qualitative and quantitative comparison of crawling methodologies and focusing on how their use can directly impact various research inferences.

More recently, we have seen works that address reproducibility concerns. Flittner et al. [52] conducted an analysis of papers from four different ACM conferences and assessed the state of artifact availability. They highlight that the majority of the analyzed papers do not provide the complete toolset required to reproduce all results and stressed upon the release of artifacts by the authors in the measurement community. Work by Scheitle et al. [75] indicates how internet top lists show considerable churn across multiple snapshots in time, impacting replicability of results. Collberg et al. [44] survey a number of computer systems papers to examine the repeatability of the experiments. They focus on their ability to locate, build, and run the system prototypes, and develop metrics that help assess the degree to which certain works are more repeatable than others. Elmenreich et al. [48] carry out an online survey to explore the need for reproducibility and provide guidelines on making research results reproducible.

7 DISCUSSION

**Limitations.** Our survey, qualitative, and quantitative studies each come with their own limitations. Some of these arise from our inability to perform large scale manual analysis and others from our inability to make observations without interfering with natural web interactions. First, due to the scale of our survey, we rely on computational filtering heuristics to narrow down the set of publications to study. This introduces the possibility of missing important insights into how crawlers are used by the academic community. Our focus on a few select conferences does not invalidate the insights provided about how certain crawlers are used and the crawl-repeatability and specificity challenges faced by the research community. Second, our qualitative analysis of crawlers broadly categorizes parameters exposed by different crawlers. This parameter generalization introduces the possibility of generalizing crawler (in)capabilities to the point of missing analysis of certain features that are important to specific communities – e.g., we do not explicitly verify the possibility of the crawler having or granting access to connected devices (e.g., cameras and microphones) – the topic of emerging studies in the privacy domain. Finally, our empirical analysis is complicated by several confounds: first, we are impacted by the several dynamic and real-time processes on the web (e.g., real-time bidding). Further, it is possible that our data has sources of bias and inconsistency since we currently only rely on IP intelligence data from Cisco’s Talos. We enable other developers with access to more threat streams to circumvent this limitation by providing an easy to use extended API. Despite these limitations, our framework replicates and in many cases pushes the boundaries of the state-of-the-art in crawling.

**Recommendations.** Despite the above limitations, our study allows us to make recommendations aimed at improving the repeatability and correctness of research relying on data generated by web crawling tools. First, we urge researchers and reviewers to enforce standards of crawl specificity to improve repeatability – e.g., at the very least, require mention of the tool, tool version, and (if applicable) browser version. Currently, over one-third of all crawler-dependant research do not mention even the name of the tool used for data gathering. Second, research communities need to understand how crawling tools might impact their inferences – particularly when inferences are drawn over multiple studies, each using a different crawling methodology. As shown in our analysis of $H_{wf}$, $H_T$ and $H_b$, this can lead to an incorrect understanding of the research domain. Therefore, we recommend that researchers aim to replicate exactly the crawling methodologies of prior work when proposing advancements to the problem domain. Alternately, there is value in evaluating results and analysis on datasets generated by multiple crawlers to ensure the correctness of inferences. To facilitate our call for repeatability and cross-crawler evaluation, we are releasing our crawling framework at https://sparta.cs.uiowa.edu/projects/methodologies.html.

**Acknowledgements.** We would like to thank our colleagues – Danyal Saeed Mirza, Hammas Bin Tanveer, Hussam Habib, John Thiede, Maaz Bin Musa, Muhammad Hassan, Muhammad Haroon, Ralph Nahra, Xiaoyu Xing, Yao Wang, and Zain Humayun – for their help in our survey. We also thank the anonymous reviewers whose valuable comments helped shaped the paper to its final form. This work was partially funded by the Spanish national grant TIN2017-88749-R (DiscoEdge) and the Region of Madrid EdgeData-CM program (P2018/TCS-4499).


