A Deeper Look at Web Content Availability and Consistency over HTTP/S

Muhammad Talha Paracha*, Balakrishnan Chandrasekaran†, David Choffnes*, Dave Levin‡
*Northeastern University, †Max-Planck-Institut für Informatik, ‡University of Maryland, College Park

***Abstract*—Recent studies of HTTPS adoption have found rapid progress towards an HTTPS-by-default web. However, these studies make two (sometimes tacit) assumptions: (1) That server-side HTTPS support can be inferred by the landing page ("/") alone, and (2) That a resource hosted over HTTP and HTTPS has the same content over both. In this paper, we empirically show that neither of these assumptions hold universally. We crawl beyond the landing page to better understand HTTPS content unavailability and inconsistency issues that remain in today's popular HTTPS-supporting websites. Our analysis shows that 1.5% of the HTTPS-supporting websites from the Alexa top 110k have at least one page available via HTTP but not HTTPS. Surprisingly, we also find 3.7% of websites with at least one URL where a server returns substantially different content over HTTP compared to HTTPS. We propose new heuristics for finding these unavailability and inconsistency issues, explore several root causes, and identify mitigation strategies. Taken together, our findings highlight that a low, but significant fraction of HTTPS-supporting websites would not function properly if browsers use HTTPS-by-default, and motivate the need for more work on automating and auditing the process of migrating to HTTPS.**

***Index Terms*—HTTPS, Internet Measurements, HTTP/S Consistency**

I. Introduction

The importance and ease of deploying HTTPS websites has increased dramatically over recent years. Recent studies of HTTPS adoption have found rapidly increasing adoption, leading some to speculate that most major websites will soon be able to redirect all HTTP requests to HTTPS [1]. Indeed, some client-side tools even go so far as to force *all* web requests to be made via HTTPS [2].

However, to our knowledge, all previous work measuring the deployment of HTTPS [1], [3]–[6] makes two basic but fundamental assumptions: (1) They assess server-side HTTPS support by looking at a single page: the domain's *landing page* (also known as its root document, "/"), and (2) They assume that any resource that is available over both HTTP and HTTPS has the same content (in the absence of an attack, of course)—that is, that the only difference between `http://URI` and `https://URI` is that the latter is served over TLS.

Unfortunately, neither of these assumptions has been empirically evaluated. This is important because, if they do not hold, they threaten our understanding of how HTTPS is truly deployed. For example, the fact that a landing page is (or is not) secure might not necessarily indicate the security of the site writ large. Moreover, if there are content differences between HTTP and HTTPS, then merely defaulting to the more secure variant—as many papers and tools have suggested—risks unexpected side-effects in usability.

978-3-903176-27-0 © 2020 IFIP

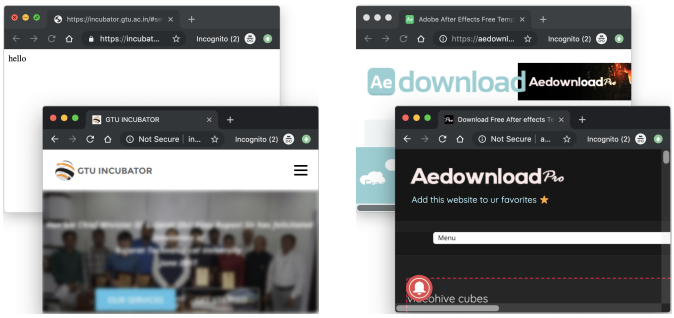


Fig. 1: Examples of webpages with different content using HTTP vs HTTPS.

In this work, we empirically show that this conventional wisdom does not universally hold. We identify inconsistencies between HTTP and HTTPS requests to the same URI, in terms of content unavailability over one protocol and content differences between them. Rather than restrict our study to landing pages, we conduct a deep crawl (up to 250 pages) of each of the Alexa top 100k sites, and for a 10k sample of the remaining 900k sites on the Alexa top 1M list [7]. We then analyze the retrieved pages to identify substantial content inconsistencies between HTTP and HTTPS versions of the same page, using a novel combination of state-of-the-art heuristics.

We find that a small but significant fraction of sites have content unavailability (1.5%) and differences (3.7%) for at least one page on the website. Fig. 1 provides two examples of how pages can differ significantly over HTTP/S. These issues affect pages across all levels of popularity, with content differences being slightly more likely on popular pages than unpopular ones. While the number of pages per site with inconsistencies is low on average, we find that such issues affect more than 50 pages for 15% of sites analyzed. We characterize common patterns for such inconsistencies, identifying that pages on websites are unavailable due to partial support of HTTPS (some directories are never served over HTTPS), different versions of webpage content available only over one protocol, and misconfigurations related to redirections and hard-coded port numbers. In addition, we find that content is different between HTTP/S for reasons such as HTTPS-support "in name only" (default server page is returned over HTTPS), redirections exclusive to one protocol, or misconfigured HTTP status codes. Last, we return to the HTTPS adoption metrics

used in prior research and evaluate their efficacy in light of our findings.

The main contributions of this paper are a large-scale measurement to reassess the consistency of HTTPS support beyond the landing page, new heuristics for automatically identifying significant content differences between HTTP/S, a characterization of observed HTTP/S inconsistencies, and analysis of mitigation opportunities. In addition to these main contributions, for the purposes of repeatability we have made our efficient web crawler code, dataset, and analysis code publicly available at https://github.com/NEU-SNS/content-differences.

II. Related Work

Relevant HTTPS attacks and mitigations There are severe security and privacy risks to users caused by a combination of HTTP-by-default behavior in web browsers, and limited deployment of HTTPS support (e.g., [8]–[10]). Prior work argues for HTTPS to be supported on every page of a website to mitigate these threats. In this paper, we investigate whether sites conform to this approach (and discover significant discrepancies).

To force a web browser to use HTTPS to access a site, web servers may use HTTP Strict Transport Security [11] (HSTS) and users can install extensions like HTTPS Everywhere in their web browser [2]. While potentially helpful, these approaches rely on manually maintained lists[1] of (non)conforming websites that cannot scale to the entire web. In light of such limitations, Sivakorn et al. [12] argue that an HTTPS-by-default approach might be the only solution. In this work, we identify that there remain a small but substantial fraction of sites with HTTPS deployment gaps (including those using HSTS) that must be addressed before the web is ready for it.

HTTPS adoption There is a significant body of work on measuring HTTPS adoption using both passive data sets and active scans (e.g., [1], [3]–[6]). Durumeric et al. [3] measured HTTPS adoption using one year of repeated scans of the entire IPv4 address space. In contrast, Felt et al. [1] conducted active scans and combined them with passive browser telemetry data—obtained from the Google Chrome and Mozilla Firefox browsers—to provide server- as well as client-based perspectives on HTTPS adoption. Overall, the results from these prior studies indicate a steady trend towards HTTPS-everywhere.

However, they either implicitly or explicitly assume that if the landing page is hosted via HTTPS then the same is true for *all* of that website's resources; and that when both HTTP and HTTPS versions of a page are available, then the content is the same across both. We can only speculate as to why these assumptions were made (and we believe they were likely done to reduce the crawling overhead compared to deep crawls with content inspection); regardless, our motivation in this work is to evaluate the impact of such assumptions.

[1]In case of HSTS, we are referring to the preload lists embedded in modern browsers. Note that HSTS through a server header is trust-on-first-use.

Kumar et al. [13] explored how third-party dependencies can hinder HTTPS adoption. In their study, 28% of HTTP-only sites were blocked from migrating to HTTPS due to their reliance on at least one HTTP-only third-party Javascript and/or CSS resource. But their analysis also assumed that resources available over both HTTP/S serve the same content, as they suggested the other 72% sites to migrate to HTTPS by merely updating the protocol in the resource URLs.

HTTPS Misconfigurations Fahl et al. [14] surveyed 755 webmasters of the sites with invalid HTTPS certificates, in an effort to determine whether the behavior was intentional or accidental. Two-thirds of the respondents admitted to deliberately using an invalid certificate, for reasons that include using a custom PKI or administering test pages not intended for real-world users. In this work, we find inconsistent content and availability for HTTP/S-supporting websites and manually investigate a sample of cases to understand their root causes.

III. Methodology

This section describes our measurement methods for crawling websites and identifying inconsistencies.

A. HTTP/S Inconsistencies

We focus our study on websites that support both HTTP and HTTPS on the landing page. For this set of sites, we measure whether there are different results when accessing the same resource over HTTP and HTTPS. We categorize such cases of *inconsistencies* between protocols as follows:

Content unavailability This occurs when a resource is successfully retrieved via HTTP, but not HTTPS (i.e., HTTP status code $>= 400$,[2] or an error[3]).

Content difference This occurs when a resource is available over both HTTP and HTTPS, but the content retrieved differs significantly (as defined in §III-C).

B. Crawling Overview

We limit our analysis to differences in HTML content served by a website, which we refer to as *pages*, and do not consider embedded resources (e.g., CSS and Javascript files). When crawling a website, we conduct a depth-first search of all links to other webpages with the same second-level (TLD+1) domain—which we call *internal links*)—starting at the landing page for the site.

The websites we crawled are a subset of the Alexa global top 1 M domains. We chose this list because it represents sites visited by users via web browsers [15]. Our analysis covers all of the top 100K most popular domains. We supplement this set with a randomly selected 10K sample of the 900K least popular domains, since rankings beyond 100K are not statistically meaningful[4]. For each domain, we crawled at most

[2]In Section IV, we show how some websites rely only on splash pages to report a failure, instead of also returning an error status codes.

[3]Except for timeout or connection-reset errors.

[4]Alexa states that they "do not receive enough data from [their] sources to make rankings beyond 100,000 statistically meaningful" [16].

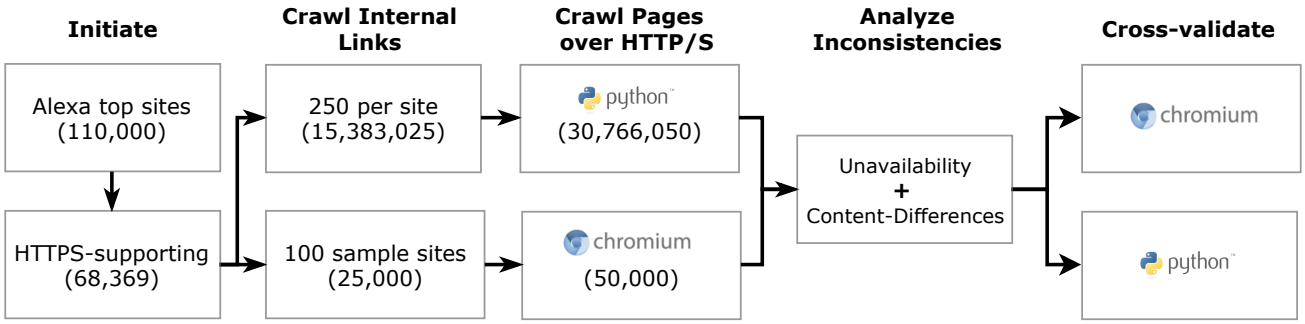


Fig. 2: Summary of our pipeline. We use a custom crawler for the full crawl, but rely on a real browser to detect any under/overestimation of inconsistencies.

250 internal links to limit the load on websites induced by our crawls while still covering significant fractions of site content. We could not identify an efficient way to inform this limit empirically, and picked this number to keep the crawl duration at an acceptable length. Note that we consider only sites where the landing page is accessible using HTTPS (and apply the same restriction to subdomains of a site).

To conduct the crawl, we developed a (i) Python-based crawler using the Requests [17], BeautifulSoup [18] and HTML5Lib [19] libraries, and a (ii) Chromium-based crawler using the Chrome DevTools Protocol. The former implementation does not attempt to render a page by executing JavaScript and/or fetching third-party resources, and is thus considerably faster, enabling us to crawl more pages in a limited set of time; we use it for finding the inconsistencies across all websites and rely on a real browser only for cross-validating our results. The crawler visited each page using both HTTP and HTTPS on their default ports (unless otherwise specified in an internal link, e.g., via a port number in the URL). A summary of the pipeline is presented in Fig. 2.

Identifying internal links To identify internal links for a site, we first access the landing page of each website[5] at the URL *http://www.<website-domain-name>*; for websites that include a subdomain, we do not add the "www." prefix. We parse the fetched webpage and retrieve all internal links (i.e., URLs from anchor tags in the page). We prune this set of internal links to include only URLs that respect the `robots.txt` file (if one exists). We also filter out URLs from subdomains according to the subdomain's `robots.txt` file. Of the 110K sites in our original list, 4,448 were filtered out due to `robots.txt` entries.

If the number of internal links retrieved from the landing page is less than the maximum number of pages per site in our crawl (250), we recursively crawl the website by following the internal links to find more URLs. We repeat this process until we have 250 URLs or we fail to find new links. Fig. 3a shows the number of pages crawled per site, with the vast majority of sites (86%) hitting the limit of 250 pages and 92% of sites yielding 50 or more pages.

[5]The *Alexa* list only provides the domain name for each website, and not the complete URL.

Ethical considerations Our crawler followed the "Good Internet Citizenship" guidelines proposed by Durumeric et al. [3]. We used a custom user-agent string with a URL pointing to our project webpage with details on the research effort and our contact information. We honored all `robots.txt` directives, carefully spread the crawl load over time, and minimized the number of crawls performed. To date, we have received only one *opt-out* request from a website administrator, which we promptly honored.

C. Identifying inconsistencies

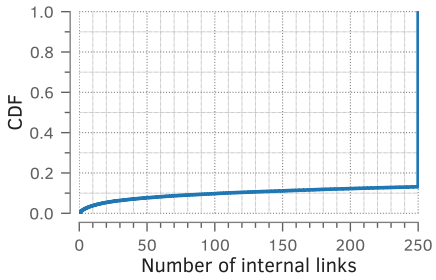
We analyze the crawled pages for inconsistencies as follows.

Identifying unavailability Despite being conservative in the rate at which we crawl web pages, it is still possible that some servers may block our requests. For example, they may blacklist the IP address of the machine running the crawler, and *not* indicate this using the standard *"429 – Too Many Requests"* response status code. Additionally, we may encounter transient *5xx* error codes. Under such circumstances, our analysis might misinterpret the temporary blocking as content unavailability (i.e., a false positive).

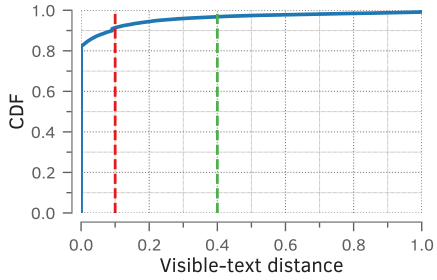
To mitigate such false positives, one day after the initial crawl we conduct a follow-up crawl of all detected cases of content unavailability. The initial crawl visited each consecutive page after the previous one finished loading, but for this smaller crawl, we use a large (20 seconds) delay between visiting consecutive pages. This follow-up crawl is designed not only to avoid rate-limiting, but also changes the order of fetches from HTTP-first to HTTPS-first to detect differences in website behavior that are dependent on order of HTTP/S access.

Our approach is susceptible to false negatives. Namely, if a server permanently blocks our crawler IP address based on just the initial crawl, the follow-up crawl would miss any potential inconsistencies for that site. This assumes that blocking is consistent for both HTTP and HTTPS requests.

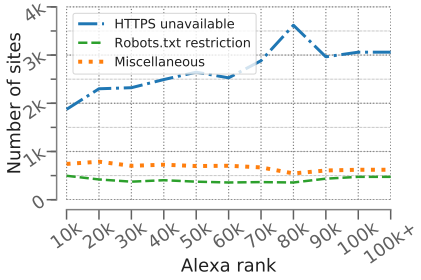
Identifying significant content differences While it is straightforward to detect pages with non-identical content using byte-wise comparisons, using this approach would flag content differences between HTTP and HTTPS for pages that are essentially identical (e.g., due to a difference in timestamp at the bottom of a page) or have dynamic nature (e.g., a product catalog page rotating featured items on each access). Thus,



(a) *Number of internal links per website that were crawled.*



(b) *CDF of visible-text distances for all byte-wise non-identical pages.*



(c) *Number of sites excluded for different reasons from the crawl.*

Fig. 3: Overview of crawling data.

to better understand meaningful content differences, we need heuristics to help filter out such cases.

In particular, we use heuristics inspired by prior research on near-duplicate detection for webpages [20], [21]. This prior work did not provide open-source implementations or sufficient justification for parameter settings, so we base our own heuristics on three steps common to all of the prior work:

- Parse HTML content and remove nonvisible parts (e.g., markup [20] and headers [21]).
- Retrieve a set of word-based n-grams[6] from the remaining content, ignoring all whitespace (n=10 [20] or n=3 [21]).
- For each pair of processed pages, compare the similarity of their n-grams.

A limitation of prior work is that it is tuned to find near-identical pages, while our goal is to find cases that are *not*. Thus, our approach is inspired by the following insights. First, we can filter out dynamic webpages by loading the same page multiple times over the same protocol—if the page content changes, then any such changes should not be counted when comparing HTTP to HTTPS. Second, we observe that dynamic pages often use the same page structure (e.g., use the same HTML template) and the changes occur only in a small set of regions in the document.

Based on these insights, we develop an algorithm for calculating "distance" between two pages, which we then use as a metric to quantify their differences on a scale of 0 to 1. First, we parse the HTML document, filtering either all text visible to a user,[7] or a list of HTML tag names (to capture the page structure). Next, we transform the result into a set of 5-grams. After getting such 5-grams for two pages, we compute the Jaccard distance (i.e., the size of the intersection of the two sets divided by the size of their union).

We then determine that there is a significant content difference between HTTP/S versions of a page if the following properties hold with parameters $\alpha, \beta, \gamma \in [0, 1]$:

1) The page-structure distance between `HTTP` and `HTTP`**`S`**, is greater than γ.
2) The visible-text distance between `HTTP` and `HTTP`**`S`**, *d-across-protocols*, is greater than α. This filters differences appearing due to minor changes such as timestamps in visible-text, and/or cookie-identifiers in the source.
3) The visible-text distance between the same page fetched twice over `HTTP` + β, is less than *d-across-protocols*. This ensures that if a page is dynamic over `HTTP`, then the difference it presents over `HTTP/`**`S`** must be greater than the baseline difference due to dynamicity, by an amount controlled by β, in order for the page to be counted in our analysis.

We obtain and use the data for computing the distances as follows. Our initial crawl loads each page over both HTTP/S to find the ones with non-identical bodies. A day later, we run a slow follow-up crawl only on the pages with non-identical bodies over HTTP/S, to identify any false positives from the initial crawl. For cases where non-identical bodies persist, we then identify pages with significant content differences satisfying the above properties. For assessing properties 1 and 2, we compare the HTTP/S versions of the page from the slower crawl. For property 3, we compare the HTTP version of the page from the initial crawl with the HTTP version of the same page from the follow-up crawl.

This method provides us with an objective way of measuring visual differences across pages served over HTTP/S, while taking into account their inherent dynamic nature. We note that whether a user actually finds a set of pages different is subjective to some extent. For the purposes of this study, we assume that the greater the visual differences across pages, the greater the probability of a user also finding them different.

Fig. 3b presents a CDF of the visible-text distances for all byte-wise non-identical sets of HTTP/S pages crawled. The majority of pages (82.4%) have a visible-text distance of 0, and are thus essentially identical. To validate this metric and determine thresholds to use for significant differences, we manually inspected more than a dozen pages at random from the set clustered around 0, and indeed find them all to have minor differences that we would *not* consider meaningful.[8]

[6]An n-gram is a contiguous sequence of n items from a given list. For example, word-based 2-grams generated from "to be or not to be" include "to be", "be or", "or not" and so on.

[7]Using the code found here: https://stackoverflow.com/a/1983219.

[8]For the sample size we used, the estimated fraction of pages with minor differences in the cluster is 0.90 $\pm$ 0.10, with a 95% confidence interval.

Type	Description	Example
Misconfigured redirections (**82.6%**)	Choosing a default protocol for all visitors, but accidentally setting up redirections which do not preserve resource paths.	*developers.foxitsoftware.cn/pdf-sdk/free-trial* gets redirected to the website homepage if the request was over HTTP (instead of being redirected to the requested page at HTTPS).
Unintentional support	Accepting HTTPS connections without serving meaningful content.	*www.historyforkids.net* presents default server page over HTTPS accesses but provides site-specific content otherwise.
Misconfigured headers	Incorrectly using the response status codes.	*www.onlinerecordbook.org/register/award-unit* returns 200 HTTP OK status for both HTTP/S accesses, but the actual content at the latter says *"Page not found! The page you're looking for doesn't exist"*.
Different versions	Providing a potentially upgraded version at HTTPS, which might have different content for the same resource request.	*video.rollingout.com* provides different content at the index page of the website based on the protocol used during request.

TABLE I: Characterization of content differences. The fraction of all pages with inconsistencies that fall into a category is reported when available.

However, there is a long tail of remaining pages with potentially significant visible-text differences—and it is possible for two pages with few visual differences to be semantically very different. The curve in Fig. 3b does not reveal any obvious thresholds for detecting significant page differences, so we provide results using a low threshold ($\alpha = 0.1$, $\beta = 0.1$, $\gamma = 0.4$) that finds more inconsistencies, and a stricter, high threshold ($\alpha = 0.4$, $\beta = 0.2$, $\gamma = 0.6$) that finds fewer. These thresholds for α are marked on the figure using vertical lines.

Although the selection of parameters entailed some manual tuning, our root cause analysis indicates that this approach worked well for identifying inconsistencies. Namely, in Section IV-A, we attribute the vast majority of the identified content differences (at least 82.6% of pages) to server misconfigurations.

D. Limitations

Our analysis is likely a lower-bound for the prevalence of inconsistencies. First, the crawler accessed only a limited number of internal pages (Fig. 3a suggests at least 86% of the websites have more than 250 pages). Second, we used a single machine for all our experiments to avoid appearing as a DDoS attack, and thus missed inconsistencies that (i) could appear due to the presence of different replicas of servers across geographical regions, (ii) would have been invisible to us due to (any) rate-limiting encountered. Third, our Python-based crawler could not execute JavaScript (JS), so we may miss inconsistencies on pages that heavily rely on it for content (we estimate this discrepancy in §IV). We finally note that the study focuses only on inconsistencies in first-party content, so we did not investigate differences in third-party dynamic content fetched via JS (e.g., inclusion of advertisements), and/or (ii) the impact of browser idiosyncrasies (e.g., HSTS preload lists, mixed-content policies [22]–[24]).

IV. Results

We performed the measurements in October 2019 from a university network in Boston, Massachusetts (USA). From the 110K sites (the *Alexa* list in §III-B) investigated, our crawler found at least one internal link for 68,369 websites available over both HTTP and HTTPS. We plot the number of sites excluded from the crawl due to various reasons, using bins of 10K websites, in Fig. 3c. The *miscellaneous* category includes cases where our crawler encountered parsing errors, pages relying on JavaScript, or a domain hosting only one page with no internal links.

For the rest, we show the number of links crawled per site as a CDF in Fig. 3a. The average number of internal links captured and crawled was 225. The cluster at $x = 250$ is due to the threshold we had set (§III-B), our crawler stops searching for new links after the limit has reached.

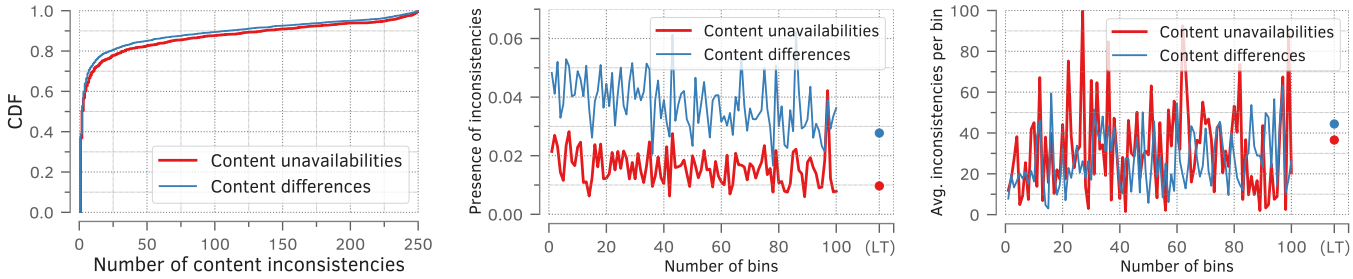
A. Summary Results

We observe 1.5% of websites (1036 of 68,369) have content unavailabilities, and 3.7% (2509 of 68,369) have content differences (3.1% with stricter significance thresholds). For websites with at least one inconsistency, the average number of pages with inconsistencies is 31.9 and 27.2, respectively. Fig. 4a plots CDFs of the number of inconsistencies per site, showing that while most sites have few inconsistencies, there is still a significant percentage of websites (15%) where we find inconsistencies for at least 50 internal links.

We identify several root causes for these inconsistencies by manually analyzing 50 random instances for both unavailability and content differences. Tables I and II provide a description and one example for each (the types represent commonly observed behaviors, and do not represent a complete taxonomy). We then label pages as (i) "Misconfigured redirections" if their final `URIs` are different over HTTP and HTTPS (i.e., after performing any redirections), and (ii) "Fixed ports" if their `URIs` include port numbers 80 or 8080. We note that it is not trivial to estimate the fractions for all categories (e.g., assessing whether a webpage presents meaningful content vs. a custom error page), and thus present the fractions only for the two above types. Most of the cases are server misconfigurations, and as such the corresponding inconsistencies are likely easy to fix (e.g., by providing suitable redirections). We note that two error types dominate content

Type	Description	Example
Misconfigured redirections (**19.7%**)	Fixing broken links through redirections, but only over one protocol.	*www.sngpl.com.pk/web/tel:1199* redirects to the website homepage when accessed with HTTP, but presents a 404 Not Found otherwise.
Fixed Ports (**8.7%**)	Embedding or enforcing port numbers in the URLs.	*facade.com/content* redirects to *facade.com:80/content/*, resulting in a connection error on accesses over HTTPS.
Partial support	Not supporting HTTPS at a portion of site content.	*www.indiana.edu/~iubpc/*, and all resources under the directory.
Different versions	Providing a potentially upgraded version at HTTPS, which might not require hosting resources from the old version.	*aedownloadpro.com/category/product-promo/* is only available at the HTTP version of the site.

TABLE II: Characterization of content unavailability issues. The fraction of all pages with inconsistencies that fall into a category is reported when available.



(a) *Number of inconsistencies per website.* (b) *Alexa rank vs. inconsistency presence.* (c) *Alexa rank vs. #inconsistencies.*

Fig. 4: Inconsistencies are prevalent on a small but significant number of websites, and are not correlated with site popularity. (LT) refers to Long Tail results.

unavailability inconsistencies; *404 Client Error* (82.6%) and *SSLError* (9.1%).[9]

Cross-validation with real browser Our results could be biased due to our reliance on a Python-based crawler that does not execute JS instead of using a real browser. This could lead to false negatives if HTTP/S differences are visible only to a JS-supporting crawler, or false positives if differences are only visible to our custom crawler.

We cross-validate our Python crawler's results by comparing them with the Chromium-based crawler, and found their results to be highly consistent. Specifically, both pipelines observed 95.8% of domains with at least one unavailability inconsistency and 86.0% of domains with at least one content difference (85.1% with stricter distance thresholds).

Upon manual inspection, we observed that domains with inconsistencies visible only via the Python-based crawler reflect (i) TLS implementation differences (e.g., minimum TLS version allowed, certificate validation logic), (ii) presence of `<meta http-equiv="refresh">`[10] redirections in page content and/or (iii) content differences that are less significant when JS is enabled. While there are differences in inconsistency results, as expected, our findings suggest that a large fraction of inconsistencies observed by the custom crawler are ones that also would affect users visiting sites via web browsers.

[9]The remaining errors are various 4xx and 5xx errors.

[10]An HTML-based instruction for web browsers to redirect to another page after a specified time.

It is also possible that the Python crawler missed inconsistencies that manifest only through a browser. To estimate whether this is the case, we used 100 sample sites in the following way. First, Alexa Top 100k list was divided into bins of size 1,000 each. From each bin, we found a sample site that had at least 250 internal pages and zero inconsistencies according to the Python-based pipeline. We fed these 100 sample sites to the Chromium-based pipeline to check for any inconsistencies that manifest only through a browser. In summary, we found no such inconsistencies.

More specifically, the browser did not flag any of these websites with unavailability issues (its usage should not have affected the availability of a page anyway), but it did find 5 with content differences (and only 1 site when using stricter significance thresholds). Upon manual analysis, we found all of these cases were due to normal web page dynamics that surpassed our content-difference significance thresholds (which were tuned for content based on a crawler that did not support JS execution).

B. Factors Influencing Inconsistencies

Website popularity We test whether inconsistency rates correlate with site popularity, and find that there is no strong relationship between the two. Specifically, we distribute entries from Alexa Top 100K list into bins of size 1,000 each, while preserving the popularity rank. Note that we use a separate bin for the 10K sample of the 900K least popular sites. In Fig. 4b, we plot bins on the x-axis and on the y-axis the fraction of websites with at least one inconsistency (the

denominator is the number of websites with at least one link crawled). In Fig. 4c, we use the same x-axis, but the y-axis is the average number of inconsistencies, for all websites with at least one inconsistency. Content differences seem slightly more likely on popular pages than unpopular ones, which is somewhat counterintuitive since one might expect more popular sites to be managed in a way that reduces content differences. But one can visually see high variance in the relationship between inconsistencies and popularity; further, we compute the Pearson's correlation coefficients and find only weak correlations—not strong enough to infer that the rate of inconsistencies depends on site popularity.

Self vs. third-party hosting Cangialosi et al. [25] studied the prevalence of outsourcing TLS management to hosting providers, as it pertains to certificate issuance and revocation. They find it to be common, and that such outsourced providers manage certificates better than self-hosted sites. Based on this observation, we investigate whether outsourced site management also reduces inconsistencies between HTTP and HTTPS.

For the case of outsourced site management, we focus on one ground-truth example: Cloudflare. We choose them because they manage certificates for the vast majority of their hosted websites.[11] For other hosting providers, it is not clear what percent of domains are being self-managed vs. service-managed. We begin by mapping a server's IP address to AS number, then use CAIDA's AS-to-Organization dataset to retrieve the organization name. We then focus on the 20,131 websites whose server organization is "Cloudflare, Inc." We find content unavailability inconsistencies in only 0.6% of these sites (a decrease of 60.0% as compared to the average across all sites), and content-differences in 2.0% (a decrease of 45.9%). Thus for this one example, Cloudflare management seems to reduce inconsistencies ($\chi^2 = 200$ with p-value < 0.00001 for Pearson's test of independence).

We compare these error rates with the set of 3,977 identified self-hosted websites. We define them as the ones whose organizations host only one domain from the Alexa 110k sites in our analysis, and thus are either self-hosted or hosted through a small provider.[12] For these, we find content unavailability inconsistencies in 4.5% of such websites (an increase of 200% as compared to the average across all sites), and content differences in 8.5% (an increase of 129.7%). Thus, self-hosted sites seem to be much more likely to have inconsistencies between HTTP and HTTPS ($\chi^2 = 226$ with p-value < 0.00001).

We posit the following reason that might explain why third-party certificate management can help reduce inconsistencies. Prior work [25] suggests third-party services perform better certificate management. They likely (i) can also notice server misconfigurations comparatively earlier due to their large number of customers and dedicated support staff, and (ii) have default TLS-related settings in place to reduce the chance of accidental mistakes when a site is migrated to HTTPS.

[11] According to a recent investor report [26], ≈97% of Cloudflare customers use the free-tier product that provides only Cloudflare-managed HTTPS certificates (consistent with estimates from prior work [25]).

[12] We manually analyzed a small sample of the 3,977 sites, and estimate the fraction of them that are self-hosted (versus hosted via a small provider) to be 0.79 ± 0.15, with a 95% confidence interval.

			Inconsistency issues	
Type	**Sources**	**Total**	**Unavailability**	**Content-diff.**
HTTPS Available	[27]–[29]	74,778	741 (1.0%)	1933 (2.6%)
	[30]	66,206	607 (0.9%)	1691 (2.6%)
Default HTTPS	[27], [30]	67,813	591 (0.9%)	1697 (2.5%)
	[28]	16,221	99 (0.6%)	368 (2.2%)
HSTS Available	[27]	17,557	108 (0.6%)	410 (2.3%)

TABLE III: Comparing HTTPS adoption metrics by calculating number of websites with issues over number of websites fulfilling the metric criteria.

Certificate issuing authority We now investigate whether the rate of inconsistencies is related to certificate issuing authority (CA). We found that across all domains crawled in the study, the most commonly used CAs are Let's Encrypt (LE; 21.6%), DigiCert (19.7%), Comodo (18.2%) and Cloudflare (8.2%). But for domains with inconsistencies, the shares change to 10.0%(↓), 31.6%(↑), 15.1%(↓) and 2.5%(↓) respectively. As such, we did not see any clear trend indicating how a CA can affect inconsistency rates.

C. Comparing HTTPS Adoption Metrics

Prior work identifies several metrics for characterizing the extent to which a website supports HTTPS [1]. In Table III, we compare our inconsistency findings with those metrics, using the Alexa list (§III-B). In some cases, our crawls identified inconsistencies on subdomains, but we exclude these from the analysis to ensure a fair comparison. We find that there exists a small but nontrivial number of sites where such metrics indicate support for HTTPS/HSTS, but we identify inconsistency issues. The results for HSTS are particularly surprising, as users are guaranteed to be affected by inconsistencies since the browser must fetch all content over HTTPS. A key takeaway is that, for a more accurate view of website HTTPS support, future scans should take into account inconsistencies and scan beyond landing pages.

D. Summary and Discussion

While our findings provide a dose of good news about the quality of HTTPS adoption on popular pages (the rate of inconsistencies is low), the sobering fact is that there are still a substantial number of inconsistencies—even on some of the most popular websites. We now discuss the implications of our findings, and why even a small number of page inconsistencies is a finding that has broad impact.

Security Prior work argues for HTTPS support on every page of a website. Thus even a single page with an unavailability issue can undermine security for all others in a site, as a single access to an insecure page (due to unavailability over HTTPS) can be a vector to enable site-wide downgrade attacks.

Usability From a usability perspective, content differences mean that a user on a HTTPS-by-default browser may view

content that the website owner did not intend to be shown. This could lead to confusion, loss of revenue (e.g., for retail sites with missing product pages or one that have incorrect details), and user abandonment. A caveat for our study is that we do not have any data regarding page popularity within a site, so we cannot tell how many users are affected by pages by content differences.

Persistency & Prevalence We found that the identified many inconsistencies are persistent over time, and the total number of issues is not getting better over time. To test this, we ran a second crawl over the Top 100K websites four months after the initial study and observed a similar-scale set of consistency issues: 1.4% of websites with unavailability issues and 3.6% of websites with content differences. Further, the *union* of all websites among the two crawls was 2% for unavailability issues and 4.8% for content-differences (with the intersection being 0.9% and 2.4% respectively). This suggests that although the issues affect a small portion of the web at *any given point of time*, the problem is much more widespread when considering the number of websites affected by it over time.

V. Conclusion

This work explored whether websites counterintuitively provide different results when the same `URI` is accessed over HTTP and HTTPS. We found a small but significant fraction of sites have inconsistency in terms of content availability and differences, and this occurs across all levels of popularity. We find that their root causes are often simple server misconfigurations, and thus recommend automated processes to identify and remediate these issues. Our findings also highlight that moving web browsers to HTTPS-by-default would still incur substantial problems for users and site accessibility, motivating the need for more study on the impact of such approaches. We argue that website administrators need tools like our crawler to help them identify and mitigate inconsistency issues to facilitate the transition to HTTPS-by-default. To encourage this, our efficient web crawler code, dataset, and analysis code can be publicly accessed at https://github.com/NEU-SNS/content-differences.

Appendix

Here we provide minor implementation details and recommendations for anyone trying to reproduce this work.

For Chromium-based pipeline, it is crucial that each page is rendered in a clean environment. We observed that using same user data directory, but launching incognito instances via *Target.createBrowserContext* command properly isolates history, cookies, HSTS directives and page cache. Our pipeline opened each URL in a headless instance and waited for the *DOMContentLoaded* event to get fired, then scrolled down the page by 500 pixels and finally retrieved response headers and page body.

While detecting any underestimation of inconsistencies, we manually analyzed the flagged URLs (12 in total) to see why they were only filtered out in the browser-based pipeline and observed that 11 were due to the dynamic nature of websites (note that JS was enabled this time; post-processing with stricter significance thresholds ruled them out), and 1 was due to a JS-based redirection which would happen after the website finished loading (causing our implementation to occasionally return HTML for the navigated page rather than original; triggering a false-alarm for content difference). Thus besides scalability issues, using a real browser for a full-crawl would also entail (i) choosing a set of different significance thresholds, and (ii) establishing a robust criterion for when a page should be considered loaded.

While detecting any overestimation of inconsistencies, we excluded a small set of originally flagged sites for whom we do not observe inconsistencies in either pipeline by the time we run the third Chromium-based verification crawl. This could either mean that site administrators fixed those inconsistencies or that we ran into temporary rate-limiting for some sites even during the slow follow-up crawl. Unfortunately, there is no way of discerning between these cases.

For future crawls, we suggest using a browser only during the internal links retrieval phase and then using the custom crawler to detect inconsistencies.

Acknowledgments

We thank Muhammad Ahmad Bashir, the anonymous reviewers and our shepherd, Ralph Holz, for their helpful feedback. This research was supported in part by NSF grants CNS-1564143, CNS-1563320 and CNS-1901325.

References

[1] A. P. Felt, R. Barnes, A. King, C. Palmer, C. Bentzel, and P. Tabriz, "Measuring HTTPS Adoption on the Web," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 1323–1338.

[2] Electronic Frontier Foundation, "HTTPS Everywhere," https://www.eff.org/https-everywhere, 2019, [Last accessed: June 17, 2019].

[3] Z. Durumeric, E. Wustrow, and J. A. Halderman, "ZMap: Fast Internet-wide Scanning and Its Security Applications," in *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*. Washington, D.C.: USENIX, 2013, pp. 605–620.

[4] J. Amann, O. Gasser, Q. Scheitle, L. Brent, G. Carle, and R. Holz, "Mission Accomplished?: HTTPS Security After Diginotar," in *Proceedings of the 2017 Internet Measurement Conference*, ser. IMC '17. New York, NY, USA: ACM, 2017, pp. 325–340.

[5] Google, "HTTPS encryption on the web – Google Transparency Report," https://transparencyreport.google.com/https/overview, 2019, [Last accessed: June 16, 2019].

[6] Electronic Frontier Foundation, "The EFF SSL Observatory," https://www.eff.org/observatory, 2010, [Last accessed: June 16, 2019].

[7] Alexa Support, "Does Alexa have a list of its top-ranked websites?" https://support.alexa.com/hc/en-us/articles/200449834-Does-Alexa-have-a-list-of-its-top-ranked-websites-, 2019, [Last accessed: June 22, 2019].

[8] M. Marlinspike, "More tricks for defeating ssl in practice," *Black Hat USA*, 2009.

[9] S. Sivakorn, I. Polakis, and A. D. Keromytis, "The cracked cookie jar: Http cookie hijacking and the exposure of private information," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 724–742.

[10] L. Chang, H.-C. Hsiao, W. Jeng, T. H.-J. Kim, and W.-H. Lin, "Security implications of redirection trail in popular websites worldwide," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 1491–1500.

[11] Internet-Draft, "RFC 6797 - HTTP Strict Transport Security (HSTS)," https://tools.ietf.org/html/rfc6797, 2012, [Last accessed: June 26, 2019].
[12] S. Sivakorn, A. D. Keromytis, and J. Polakis, "That's the way the cookie crumbles: Evaluating https enforcing mechanisms," in *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*. ACM, 2016, pp. 71–81.
[13] D. Kumar, Z. Ma, Z. Durumeric, A. Mirian, J. Mason, J. A. Halderman, and M. Bailey, "Security challenges in an increasingly tangled web," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 677–684.
[14] S. Fahl, Y. Acar, H. Perl, and M. Smith, "Why eve and mallory (also) love webmasters: a study on the root causes of ssl misconfigurations," in *Proceedings of the 9th ACM symposium on Information, computer and communications security*. ACM, 2014, pp. 507–512.
[15] Q. Scheitle, O. Hohlfeld, J. Gamba, J. Jelten, T. Zimmermann, S. D. Strowes, and N. Vallina-Rodriguez, "A Long Way to the Top: Significance, Structure, and Stability of Internet Top Lists," in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC '18. New York, NY, USA: ACM, 2018, pp. 478–493.
[16] Alexa Support, "How are Alexa's traffic rankings determined?" https://support.alexa.com/hc/en-us/articles/200449744-How-are-Alexa-s-traffic-rankings-determined-, 2019, [Last accessed: June 22, 2019].
[17] K. Reiz, "Requests III: HTTP for Humans and Machines, alike." https://3.python-requests.org, 2018, [Last accessed: June 17, 2019].
[18] L. Richardson, "PyPI: beautifulsoup4," https://pypi.org/project/beautifulsoup4/, 2019, [Last accessed: June 17, 2019].
[19] J. Graham, "PyPI: html5lib," https://pypi.org/project/html5lib/, 2017, [Last accessed: June 17, 2019].
[20] M. Henzinger, "Finding near-duplicate web pages: a large-scale evaluation of algorithms," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2006, pp. 284–291.
[21] Moz Developer Blog, "Near-duplicate Detection at Moz," https://moz.com/devblog/near-duplicate-detection/, 2015, [Last accessed: June 17, 2019].
[22] Mozilla, "MDN web docs: Firefox 23 for developers," https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Releases/23, 2013, [Last accessed: June 17, 2019].
[23] C. Evans and T. Sepez, "Google Security Blog: Trying to end mixed scripting vulnerabilities," https://security.googleblog.com/2011/06/trying-to-end-mixed-scripting.html, 2011, [Last accessed: June 17, 2019].
[24] IEBlog, "Internet Explorer 9 Security Part 4: Protecting Consumers from Malicious Mixed Content," https://blogs.msdn.microsoft.com/ie/2011/06/23/internet-explorer-9-security-part-4-protecting-consumers-from-malicious-mixed-content/, 2011, [Last accessed: June 17, 2019].
[25] F. Cangialosi, T. Chung, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "Measurement and analysis of private key sharing in the https ecosystem," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 628–640.
[26] Cloudflare, "Investor Presentation Q4 2019," https://cloudflare.net/files/doc_downloads/Presentations/2020/NET-Q4-2019-Investor-Presentation_FINAL.pdf, 2020, [Last accessed: May 10, 2020].
[27] Mozilla, "Mozilla Observatory," https://observatory.mozilla.org/, 2017, [Last accessed: June 26, 2019].
[28] HTTPSWatch, "About," https://httpswatch.com/about, 2017, [Last accessed: June 26, 2019].
[29] Censys, "Censys," https://censys.io/, 2017, [Last accessed: June 26, 2019].
[30] Google, "HTTPS FAQs," https://support.google.com/transparencyreport/answer/7381231/, 2017, [Last accessed: June 26, 2019].