Measuring HTTPS Adoption on the Web

Adrienne Porter Felt, *Google;* Richard Barnes, *Cisco;* April King, *Mozilla;*
Chris Palmer, Chris Bentzel, and Parisa Tabriz, *Google*

https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/felt

This paper is included in the Proceedings of the
26th USENIX Security Symposium

August 16–18, 2017 • Vancouver, BC, Canada

ISBN 978-1-931971-40-9

Measuring HTTPS Adoption on the Web

Adrienne Porter Felt[1], Richard Barnes[2], April King[3], Chris Palmer[1], Chris Bentzel[1], Parisa Tabriz[1]
[1]*Google,* [2]*Cisco,* [3]*Mozilla*
[1]*felt, palmer, cbentzel, parisa@chromium.org,* [2]*rlb@ipv.sx,* [3]*april@mozilla.com*

Abstract

HTTPS ensures that the Web has a base level of privacy and integrity. Security engineers, researchers, and browser vendors have long worked to spread HTTPS to as much of the Web as possible via outreach efforts, developer tools, and browser changes. How much progress have we made toward this goal of widespread HTTPS adoption? We gather metrics to benchmark the status and progress of HTTPS adoption on the Web in 2017. To evaluate HTTPS adoption from a user perspective, we collect large-scale, aggregate user metrics from two major browsers (Google Chrome and Mozilla Firefox). To measure HTTPS adoption from a Web developer perspective, we survey server support for HTTPS among top and long-tail websites. We draw on these metrics to gain insight into the current state of the HTTPS ecosystem.

1 Introduction

For most of the Internet's history, HTTP Web traffic traveled unencrypted between clients and servers. After widespread tampering and surveillance in transit came to public attention (e.g., [20, 26, 28]), cross-industry efforts arose to promote the use of HTTP over TLS (HTTPS). In response, many large websites transitioned to serve HTTPS by default (e.g., [25, 33, 34]).

How much of the web is currently HTTPS, and are adoption rates trending positively? We want to understand the growth of HTTPS for two reasons:

- Security engineers and researchers have put significant effort into projects like Let's Encrypt,[1] The HTTPS-Only Standard,[2] and search ranking changes [7] to promote HTTPS. HTTPS adoption metrics allow us to see whether these combined efforts have succeeded at shifting the Web at large. Is there more work to do (and if so, where)?

- To protect users, browsers now require HTTPS for certain Web features and UI treatments (e.g., [1, 24, 32, 3]). They plan to make further changes as HTTPS becomes the default standard [30, 3]. How close is the Web to considering HTTPS a default?

In this paper, we measure HTTPS adoption rates from the perspectives of both clients and servers. A key challenge is that there are many ways to measure client usage and server support of HTTPS, each yielding different findings on the prevalence of HTTPS. For example, HTTPS is a much higher fraction of browser page loads if the metrics count certain types of in-page navigations. We address this challenge by examining HTTPS adoption from several angles, surveying a broad set of HTTPS adoption metrics and discussing the considerations of each. This yields a holistic picture of HTTPS adoption.

To understand the user experience of HTTPS, we measured the browsing habits of Chrome and Firefox clients at scale using several browser telemetry metrics. However, browser statistics are weighted towards the larger websites that make up a greater proportion of traffic, and a healthy Web ecosystem also encourages the participation of small- and medium-sized Web developers. We therefore also scanned large sets of servers to see whether they support HTTPS by default, not by default, or not at all. We use publicly available Web scanners and their data sets alongside our own tools and data. Finally, we examined publicly-available data on network traffic volumes at one Internet backbone provider to get a sense of how much web traffic is HTTPS in aggregate.

We find that HTTPS adoption grew substantially over the last few years. A majority of browsing is now done over HTTPS on desktop, having increased by more than ten points in 2016 alone. The number of top websites serving HTTPS by default doubled between early 2016 and early 2017. However, significant work still remains as of February 2017. Half of top websites are still HTTP by default, and most servers in the long tail don't support HTTPS at all. Mobile Web browsing lags behind desktop, and East Asian countries have substantially lower HTTPS usage rates than the rest of the world.

Contributions. We contribute the following:

- We present a holistic view of HTTPS adoption by examining data from many vantage points. We collected large-scale client data from two major browsers, curated and scanned lists of websites, and surveyed other publicly available data sets.
- We evaluate how HTTPS adoption has grown over time, from both client and server perspectives.
- We investigate factors that influence HTTPS adoption rates, including website popularity (top websites vs the long tail), the client's country, and the client's operating system.
- We identify areas where outreach and investigation could have high impact on the HTTPS ecosystem.
- We show that a single metric does not capture HTTPS adoption, discuss why, and provide guidance on when to use different metrics.

2 Background

2.1 What is HTTPS?

> *"...the Web's trustworthiness has become critical to its success. If a person cannot trust that they are communicating with the party they intend, they can't use the Web to shop safely; if they cannot be assured that Web-delivered news isn't modified in transit, they won't trust it as much." [30]*

HTTPS [31] is the secure variant of the HTTP protocol [18] on which the Web is based. HTTPS provides cryptographic security protections by carrying HTTP messages over the Transport Layer Security protocol instead of directly over TCP [9]. Websites are authenticated using digital certificates [8]. In the Web context, browsers also enforce additional policies for HTTPS pages, for example ensuring that HTTPS pages cannot load scripts from non-secure sources [37].

Together, these mechanisms protect Web traffic from network attackers in a few ways:

- **Confidentiality:** Communications between the browser and the web server are not accessible in plaintext to intermediate entities.
- **Integrity:** Intermediate entities cannot make modifications to content sent between the browser and the web server.
- **Server authentication:** The client is assured that the other end of the channel is the one that it intends to communicate with.

Data that are not protected by TLS are also not protected by HTTPS. For example, a network attacker may observe the lengths of TLS records (from which certain features can be inferred [36][22]), or the server name sent in the TLS Server Name Indication extension [13].

HTTPS is focused on protection against network attackers, and does not provide protections against other classes of attacker. For example, it is still possible for a web-level attacker to launch Cross Site Scripting (XSS) attacks against HTTPS websites; to protect against these attacks, a website would need to deploy mechanisms such as Content Security Policy [35]. These mechanisms, however, are dependent on HTTPS to be robust against attacks at the network layer.

Web servers can support HTTPS, HTTP (without TLS), or both. Typically, clients reach web servers over HTTPS by using URLs beginning with the `https:` scheme. We say that a site supports HTTPS "by default" when requests to URLs with the non-secure `http:` scheme are redirected to `https:` URLs. This can be done, for example, with the HTTP 301 status code or HTTP Strict Transport Security [19].

2.2 HTTPS promotion efforts

The security community has invested significant effort into evangelizing, supporting, and requiring HTTPS adoption. A few examples of related projects are:

- Let's Encrypt, "a free, automated, and open Certificate Authority," aims to make certificate provisioning easier for Web developers.[1] As of January 2017, Let's Encrypt supported more than 20,000,000 active certificates [4].
- Google search ranking uses HTTPS support as "a very lightweight signal" [7]. In theory, this encourages ranking-conscious websites to adopt HTTPS.
- In early 2017, Mozilla Firefox and Google Chrome began warning users against entering passwords and credit cards on HTTP websites [3, 32].
- Qualys SSL Labs built an online testing tool that "performs a deep analysis of the configuration of any SSL web server on the public Internet."[3] It is widely used to test TLS configurations.
- Google Chrome added a new Security Panel to help developers debug issues with HTTPS [2].
- Technologists within the United States federal government are moving government websites to HTTPS en masse. The White House Office of Management and Budget issued a memorandum requiring HTTPS for federal websites.[2]

- Google's Transparency Report tracks HTTPS adoption across Google products and popular non-Google websites. Their goal "is to hold ourselves accountable and encourage others to encrypt" [17].

2.3 Related work

Network-based measurements. Several studies have addressed the usage and quality of TLS and HTTPS from the perspective of the network, both by way of scans of the IPv4 address space [12, 27] or by analyzing traffic captured from network links [21, 29]. These projects measure HTTPS adoption at a very broad scale. (For example, they don't distinguish between top-level page loads and subresource requests.) This paper combines browser telemetry, scans, and network-based measurements to form a more holistic view of HTTPS usage.

Alexa Million scans. Durumeric et al. [11] scanned the Alexa Top Million repeatedly from 2012 to 2013, observing a 23% increase in the number of Alexa Top Million websites serving certificates during this time period. We continue this line of work (now updated for 2017) and complement it with additional metrics.

HTTPS errors. Webmasters often misconfigure HTTPS on their websites, either intentionally or accidentally. Several large-scale measurement studies have examined this issue. Akhawe et al. measured the frequency of HTTPS errors from a network perspective, finding that 1.54% of 3.9 billion TLS connections resulted in errors [5]. Follow-up measurement studies looked at the causes of the errors [14] and users' reactions to them [6, 15]. Our work focuses on the broader question of how often HTTPS is used at all.

Blog posts. We previously shared some of our metrics in public blog posts[4] and the Google Transparency Report. This paper gathers the metrics into a single place and adjusts them to be as comparable as possible. This paper also provides an in-depth discussion of our methods and implications of the metrics, which were lacking from the high-level blog posts.

3 Client usage of HTTPS

We aim to measure how much Web browsing happens over HTTPS on end-user devices. To this end, we use browser telemetry in Mozilla Firefox and Google Chrome to measure client usage of HTTPS at scale.

3.1 Browser telemetry background

Google Chrome and Mozilla Firefox have similar user metrics programs, referred to as *telemetry*. We use telemetry to study HTTPS usage statistics over a significant portion of the overall browser user base.

Types of data. Browser telemetry collects metrics in the form of enums, times, and booleans. The metrics are tagged by the client's operating system, client's country, and an opaque identifier for the client. One intentional limitation is that they do **not** include user characteristics like age, gender, or occupation to protect user privacy.

Computation. All of our HTTP(S) telemetry metrics are computed wholly on the client side. When pages are opened or closed, we record the HTTP(S)-related event by incrementing the appropriate histogram or time variable. Thus, we only transmit histograms and floating-point numbers to the server.

Optional participation. Telemetry is optional, with controls available in browser settings. The release version of Firefox is opt-in, with 0.7% of release users opted in. Chrome telemetry is opt-out (it was opt-in prior to Chrome 54). A much larger fraction of users participate in Chrome's telemetry program, amounting to billions of page load events in our Chrome data set. Pre-release versions of both browsers are opt-out.

Non-identifiable. Telemetry metrics are meant to be consumed at scale. Our metrics do **not** include any personally identifiable fields or browsing history.

Browser channel. Most people run the release (i.e., "stable") channel version of their browser. A small number of people use pre-release versions in order to see new browser features early and/or provide feedback to browser vendors. We report telemetry data only from the browsers' release channels because it has more external validity thanks to its "typical" users.

3.2 Metric definitions

We examine four metrics: two page load metrics, one time-based metric, and one transaction-based metric. When designing the metrics, we faced the challenge of flattening a qualitative user experience (browsing) into a quantitative metric (percentage of an event).

3.2.1 Page load metrics

Browser vendors often measure feature usage by page load – "what percentage of page loads use feature X?" – to estimate how often users encounter a feature. We accordingly measured HTTPS usage by page load, using two page load-based metrics.

Our primary metric is the **strict page load metric**. Every time a top-level page finishes loading, we record the protocol in a histogram.[5] We restrict the metric to page navigations to successful website page loads by excluding non-HTTP(S) protocols like `chrome://`, browser error pages, the New Tab Page, History API navigations, and fragment navigations.

The metric is implemented similarly in Chrome and Firefox, with two differences. First, Chrome and Firefox treat navigations to cached pages differently. The Firefox version of the metric ignores cached pages except in case of cache revalidation. Second, Chrome excludes non-HTML resources (like PDFs) but Firefox includes them. These are implementation artifacts due to the different navigation metric hooks available in each browser.

Our secondary page load metric is the **extended page load metric**, which we record in Chrome.[6] It is identical to the strict page load metric except it also counts in-page navigations. An *in-page navigation* is when a website uses the History API or URL fragments to navigate; this changes the visible URL but does not actually load a new page. Several of the Internet's most popular websites make heavy use of in-page navigations. E.g., Facebook dynamically swaps out page content when someone clicks on a friend's name in the News Feed, using the History API to make it look like the URL changed. Browsing Facebook for an hour generates a large number of extended page loads but only one strict page load. This technique is not commonly used in the long tail of the Internet because it is significantly more technically complex than typical link-based site navigation.

Both page load metrics are sensitive to whether and how people make use of tabs. Consider Alice and Bob both searching for "cats" on Google:

1. *Tabbed window navigation.* Alice opens the first search result in a new tab. When she's done, she goes back to the Google tab and opens the next search result in a new tab. She repeats this for the first nine search results. Consequently, 10% of her page loads were over HTTPS: one Google tab over HTTPS and nine HTTP search result tabs.

2. *Single window navigation.* Bob opens the first search result in his main browser window. When he's done, he hits the "Back" button to return to Google. He repeats this for the first nine search results. Consequently, 50% of his page loads were over HTTPS: Google over HTTPS, HTTP search result, Google, search result, Google, search result...

Alice and Bob saw the same exact websites in the same exact order, but they generated very different page load metrics (10% vs 50%).

3.2.2 Time in foreground

In Chrome, the **time in foreground metric** measures how much wall clock time people spend looking at websites, and whether those websites are HTTP or HTTPS. We added this metric after we grew concerned about the effect of tabbed browsing on page load metrics.

Every time a top-level page is closed, we record the protocol and the amount of time that the page spent in the foreground. Like the page load metrics, we exclude non-HTML resources, non-HTTP(S) protocols, incomplete navigations, and the New Tab Page. It *does* include time spent on cached websites. In-page navigations are irrelevant to this metric because all time spent on a given protocol is summed together.

3.2.3 Transactions

In Firefox, we record the percentage of HTTP transactions that occur over HTTP or HTTPS. The **transaction metric** is implemented similarly to the strict page load metric, but it counts HTTP transactions instead.[7]

The transaction metric is the least likely to reflect the user experience of web browsing. The transaction-based metric is sensitive to hidden implementation details of websites because it includes both top-level page loads and subresource requests. A single page load might issue anywhere from zero to hundreds of resource requests. For example, the Washington Post homepage issues 262 requests to 41 origins. Consider someone who opens two websites — one HTTP and one HTTPS — and spends equal amounts of time on them. Intuitively, this scenario ought to yield a 50% HTTPS usage rate. However, if the HTTP page generated one request and the HTTPS page generated nine requests, the transaction metric would record a 90% HTTPS usage rate.

Despite this metric's limitations, we nonetheless feel this is a useful metric to record and share for reference. The transaction metric is the most similar to network-based HTTPS metrics, which cannot distinguish between top-level page loads and subresources.

3.3 Results

As of February 2017, HTTPS comprises a majority of browsing in Mozilla Firefox and Google Chrome (on desktop). HTTPS usage lags behind on Android in Chrome by the extended page load and time-in-foreground metrics. We also find that HTTPS usage differs globally, with East Asian countries exhibiting markedly lower HTTPS usage rates. Overall, HTTPS usage rates continue to rise over time.

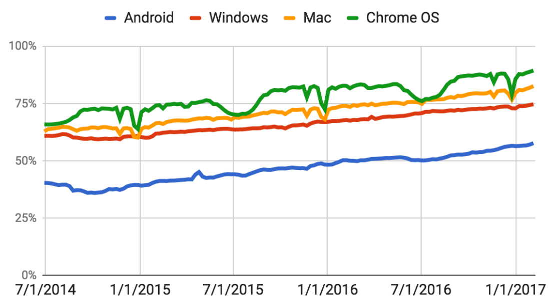

Figure 1: The percentage of extended page loads over HTTPS from July 2014 to February 2017, in Chrome.

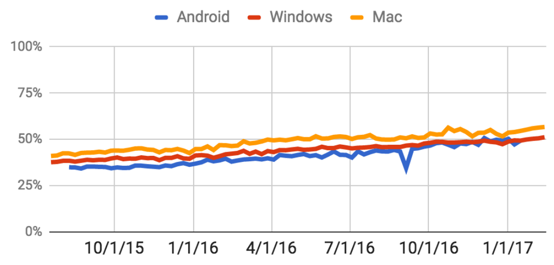

Figure 2: The percentage of strict page loads over HTTPS from July 2015 to February 2017, in Firefox.

3.3.1 Usage over time

Is HTTPS usage still growing? (Yes.)

Given the security community's investments into HTTPS adoption, we hope to see sustained growth in HTTPS usage. At present, HTTPS usage is still growing. Figure 1 shows extended page load metrics in Chrome from July 2014 to February 2017, and Figure 2 shows strict page load metrics in Firefox from July 2015 to February 2017. Both demonstrate consistent (albeit non-monotonic) growth. As of Febrary 2017, HTTPS usage continues to increase overall for clients of both browsers despite short-term fluctuations.

We observe that HTTPS usage has weekly and seasonal variations. As shown in Figure 4, there is more HTTPS usage Monday through Friday than on Saturday and Sunday. (Recall that HTTPS usage is a *percentage* — this does not necessarily indicate a decrease in overall browsing over the weekends.) A chi-squared test on the cross-tabulation of HTTP/HTTPS vs. weekday/weekend shows this difference to be significant with very high confidence ($p < 10^{-15}$). We hypothesize that work- and school-oriented websites are more likely to be accessed over HTTPS than leisure websites, and users' browsing habits shift between these depending on the day of the week. The timelines also show that holidays correlate with temporary HTTPS usage drops for some classes of users. For example, HTTPS usage on Mac and Chrome OS drops during Christmas and New Years. In addition to the winter holidays, HTTPS usage on Chrome OS also dips during the northern hemisphere's summer. We hypothesize that this is also due to differences between work and leisure browsing, and it is especially pronounced in Chrome OS due to Chrome OS's popularity in North American schools.

3.3.2 Client operating system

Does HTTPS usage differ by operating system? (Yes.)

Figure 3 shows our browser HTTPS metrics, split by client operating system. HTTPS comprises a majority of browsing on all three desktop operating systems and Firefox for Android by all available metrics, as of February 6, 2017. However, Chrome for Android users spend more time on HTTP than on HTTPS.

Due to the differences between client operating systems, a single summary statistic across all types of clients would be misleading. Such a statistic would overstate HTTPS usage on Android and understate HTTPS usage on desktop. Further, it would be sensitive to shifts in computing trends; for example, a decrease in Android phone usage would make the overall HTTPS usage rate appear to increase. We therefore split our statistics by operating system, focusing on Windows and Android as the largest populations.

Android. HTTPS usage is lower on Android than other operating systems. The difference is largest in Chrome, where less than half of strict page loads and time in foreground are spent on HTTPS websites. The gap between Android and desktop is smaller among Firefox users, but Android still has the lowest HTTPS usage rates among Firefox platforms.

We hypothesize that lower HTTPS usage rates on Android are due primarily to the popularity of native Android apps like Facebook, Twitter, and Google Search, in place of the equivalent web apps. Browser metrics can't capture search, e-mail, or social media when they are not in the browser. App usage is "invisible" from the browser's perspective, and app usage is concentrated in a small number of popular apps.[8] This leaves Android web browsing more tail-heavy than other operating systems.

The difference between mobile and desktop browsers might also be related to the types of sites users are visiting. If the hypothesis that work-related sites have more HTTPS than leisure sites is valid, then the difference between mobile and desktop might be a result of users tending to visit more leisure sites than work sites on their mobile devices, and vice versa on their desktop computers.

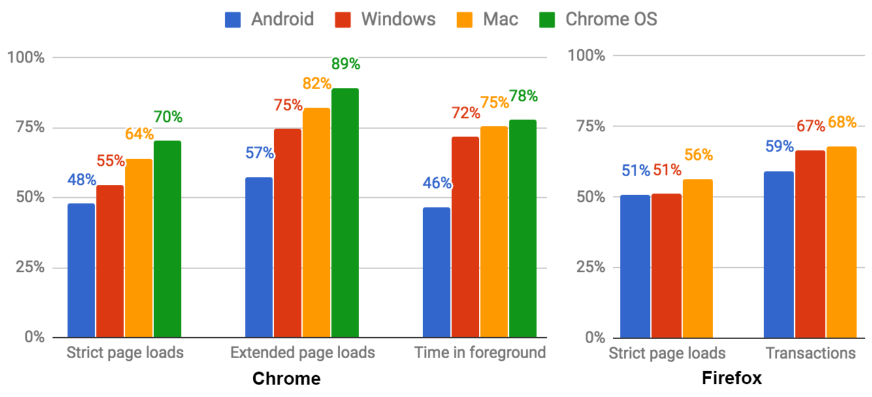

Figure 3: Browser HTTPS usage metrics for the week ending February 6, 2017. (Firefox for Android metrics are for the week ending January 23, 2017 due to a data processing issue.)

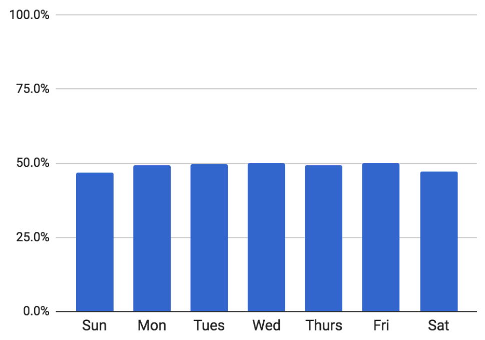

Figure 4: The percentage of strict page loads over HTTPS in December 2016, in Firefox, by day of week.

Desktop operating systems. Windows, Mac, and Chrome OS clients display similar HTTPS usage patterns, with Windows having the lowest HTTPS usage rates and Chrome OS having the highest. The success of HTTPS on Chrome OS might be influenced by demographics: Chrome OS users might be more likely to use other Google products, which are HTTPS by default.

3.3.3 Regional disparities

Is HTTPS usage equal across the world? (No.)

Web browsers serve global audiences, whose cultures and browsing habits differ. This could yield different HTTPS usage rates. Figure 5 shows a global view of HTTPS usage among Firefox users. Table 1 shows metrics for a subset of countries in Chrome, selected for cultural diversity and large Internet-using populations.

Emerging markets. We initially expected that emerging markets would have lower HTTPS usage rates, but we do not see evidence of that in our metrics. For example, India's rates are slightly higher than Germany's by all measures, and Brazil's are slightly higher than France's by several metrics.

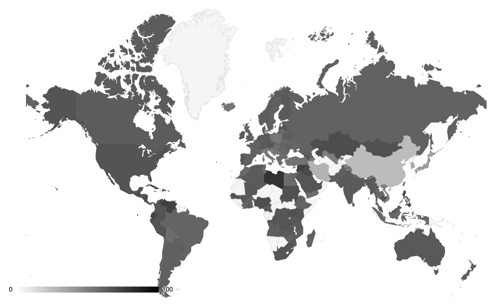

Figure 5: A map showing median rate of HTTPS usage among Firefox users by country from February 5–8, 2017, excluding countries with small user populations.

High HTTPS usage. Small countries have the highest HTTPS usage rates. In Chrome, 90% of strict page loads in Tuvalu, Svalbard and Jan Mayem, and Benin are HTTPS. In Firefox, the 75th percentile HTTPS usage rate is above 90% in Mayotte, Libya, Syria, Venezuela, Ecuador, and Iraq, i.e., 25% of users in these countries use HTTPS on more than 90% of page loads. We hypothesize that browser users in small countries spend more time on large, centralized websites like Google and Facebook (which support HTTPS by default) due to the lack of localized long-tail Web content. It is also possible that, due to the small sample sizes, certain groups of "unusual" users skew the statistics in these countries.

Most large countries have similar HTTPS usage rates, within a 10-point range. However, within this range, the United States is a consistently high consumer of HTTPS across all metrics. India and Mexico are close behind.

Low HTTPS usage. East Asian countries are notable outliers. South Korea, Japan, and China have very low HTTPS usage rates, lagging far behind other countries. The only other country outside of East Asia with similarly low HTTPS usage is Iran, which is second only to

		Chrome Android			Chrome Windows			Firefox
		SPL	*EPL*	*Time*	*SPL*	*EPL*	*Time*	*Median SPL*
Brazil	BR	51%	63%	53%	55%	77%	73%	61%
Canada	CA	53%	60%	49%	58%	76%	74%	64%
France	FR	51%	59%	46%	56%	71%	66%	61%
Germany	DE	52%	58%	48%	56%	71%	68%	64%
India	IN	54%	60%	51%	58%	74%	70%	65%
Indonesia	ID	46%	58%	47%	50%	73%	68%	59%
Japan	JP	26%	32%	24%	36%	55%	52%	37%
Mexico	MX	51%	63%	51%	57%	82%	77%	66%
Russia	RU	50%	69%	50%	55%	81%	72%	61%
South Korea	KR	34%	35%	29%	29%	44%	44%	33%
Spain	ES	49%	58%	44%	53%	73%	66%	62%
Turkey	TR	49%	54%	41%	48%	69%	63%	55%
United States	US	57%	63%	55%	63%	77%	76%	67%

Table 1: Strict page load (SPL), extended page load (EPL), and time-in-foreground metrics for thirteen countries with large Internet-using populations. Chrome metrics are for the week ending on February 6, 2017. For Firefox, the median SPL rate among users in each country over the period February 5–8, 2017.

China in terms of median HTTPS usage among Firefox users. The disparity between these countries and the rest of the world highlights the challenge of creating global browser policies when people in some countries have very different browsing experiences.

Efforts to increase HTTPS usage in East Asia would have both local and global benefits. Not only would it increase Internet privacy in those countries, but it would also allow browser vendors to move more aggressively on HTTPS-preferential policies. To start, we recommend investigating *why* Japanese and South Korean browser users are less likely to use HTTPS. Although low HTTPS usage rates in China might be due to the Great Firewall, the same cannot be said for Japan and South Korea. Is it cultural (e.g., less concern about privacy), technical (e.g., legacy infrastructure in popular East Asian websites), or legal (e.g., different laws regarding cryptography)? Once the stumbling blocks are understood, outreach efforts to popular websites in East Asia could target those hurdles.

4 Server support for HTTPS

We cannot look at HTTPS adoption from only the browser perspective. Browser statistics are weighted towards the larger websites that make up a greater proportion of traffic, and we also wish to understand HTTPS adoption among small- and medium-sized websites in the "long tail" of the Web.

To this end, we scan lists of websites to measure HTTPS support across the web. We want to know how many websites (a) support HTTPS at all or (b) offer HTTPS by default. From a methodological standpoint, scanning websites for HTTPS support has two components: the testing technique (how did we determine HTTPS support?), and the list of websites (how do we define "the web"?). We use our own testing tools and lists alongside public tools and lists.

4.1 Testing tools

We want to measure whether a server supports HTTPS. In recent years, several competing tools (including two of our own) have arisen to perform this task using slightly different sets of criteria. We survey these tools and attempt to use them in a comparable fashion.

4.1.1 Mozilla Observatory

Mozilla created the Mozilla Observatory[9] as a tool to test servers' HTTPS configurations, along with a few other security properties. The Mozilla Observatory performs a handful of simple tests to determine whether and how a website is accessible over HTTPS. The results of the scans are publicly available via an API.

HTTPS available. The most basic test is to connect to the domain on port 443 and make a HTTPS request for the root document. If the website returns a valid certificate and the request succeeds – regardless of redirections or status codes – we consider it available over HTTPS.

Default HTTPS. This test assesses whether a website forcibly redirects HTTP traffic to an HTTPS endpoint.

HSTS. When a website sets a HTTP Strict Transport Security (HSTS) header, browsers that support the header will always use HTTPS to connect to that website. We test whether a website provides the HSTS header with a minimum max-age of six months (the minimum required for preloading at the time the tool was built).

HSTS preloading. A man-in-the-middle attacker can prevent a client from receiving a website's HSTS header. To stop this attack, several browsers use a preloaded list of websites that serve HSTS headers. We check whether each website appears on the preloaded list.

HPKP. If a man-in-the-middle attacker were to collude with a rogue CA, the attacker could present a forged certificate that *appears* legitimate. HTTP Public-Key-Pinning (HPKP) ensures that the browser will only accept specific certificates intended by the website. The Observatory tests whether the HPKP header is implemented for a given website, with any max-age.

4.1.2 Google Transparency Report

The Google Transparency Report[10] scans a set of 100 non-Google websites.[11] The results are updated weekly. We maintain the Transparency Report's pre-existing testing infrastructure to track two criteria over time.

HTTPS available. A website is considered to work on HTTPS "if the Googlebot successfully reaches https://domain and isn't redirected through an HTTP location". The server must provide a valid HTTPS certificate chain for the website. This is more stringent than the "HTTPS available" category as defined by the other tools (Mozilla Observatory, HTTPSWatch, and Censys), which all permit redirects through HTTP.

Default HTTPS. A website is considered HTTPS by default if "the site redirects HTTP requests to a HTTPS URL" in response to a connection attempt from the Googlebot. The server must provide a valid certificate chain for the website. Per their rating system, a website does not need to use HSTS to achieve this designation.

Counter-intuitively, a website can be HTTPS by default without making HTTPS available. This situation occurs most notably with subdomains. For example, "http://domain redirects to https://subdomain.domain, but https://domain refuses the connection".

4.1.3 HTTPSWatch

HTTPSWatch tests prominent websites for HTTPS support.[12] We place websites into two categories based on HTTPSWatch's three-tier rating system. (The authors of this paper are not involved in the HTTPSWatch project; we use it as a public repository of HTTPS data.)

HTTPS available. We assign a website to this category if HTTPSWatch's client server can establish a verified TLS connection to the website. Our label corresponds to either the "Mediocre" or "Good" ratings in HTTPSWatch's published rating system.

Default HTTPS. A website is considered to support HTTPS by default if a verified TLS connection can be established, the HTTP version of the website redirects to HTTPS, and the HSTS header is set. Our label corresponds to the "Good" rating in HTTPSWatch's rating system. This is comparable to the Mozilla Observatory's "HSTS" category and is more strict than the Google Transparency Report's "default HTTPS" category.

4.1.4 Censys

Censys "is a public search engine that enables researchers to quickly ask questions about the hosts and networks that compose the Internet"[13]. It maintains a large database of server configurations, including information about TLS support [10]. We query Censys to test whether servers support HTTPS. (The authors of this paper are not involved in the Censys project; we use it as a public repository of HTTPS data.)

HTTPS available. A server is considered to support HTTPS if it responds on port 443 and provides a valid certificate chain [10]. In Censys syntax, this corresponds to `443.https.tls.validation.browser_trusted: true and protocols: "443/https" and 443.https.tls.validation.matches_domain: true"` for websites. Censys cannot enforce the last restriction for an IPv4 host, so an IPv4 host is considered to support HTTPS even if the certificate might fail to validate in a browser due to a name mismatch error.

4.2 Lists of websites

Different lists of websites are useful for different research questions. Do we care about HTTPS adoption for the whole Internet, popular websites, or websites popular in India? We use several publicly available lists of websites, each of which has its own characteristics. See Appendix A for copies of the lists.

HTTPSWatch Global. The HTTPSWatch project provides a list of 40 "prominent websites" in five areas: search, social media, commerce, cloud storage, and publishing platforms. The list was curated by the project to represent well-known, influential websites in each area. They describe their selection criteria as, "HTTPSWatch's goal is to list several representative sites for each category. Usually these are the most popular sites, so HTTPS

List	List size	Tool	HTTPS available	Default HTTPS
HTTPSWatch Global	40	HTTPSWatch	80%	35%
Google Top 100	100	Googlebot	54%	44%
Alexa Top 100 Global	100	Mozilla Observatory	87%	23%
Alexa Million	969,278	Mozilla Observatory	40%	10%
Alexa Million	856,312	Censys	38%	N/A
IPv4 hosts	101,052,620	Censys	10%	N/A

Table 2: HTTPS support among each set of websites, February 2017.

support on them affects the most users."[12] We recorded the state of HTTPSWatch on February 13, 2017.

The project also maintains country-specific lists, but we do not report them here. Although their country-specific lists have value, they are unsuitable for comparing HTTPS adoption rates across countries because each country's list has different categories and criteria.

Alexa Top 100. The Alexa Top 100 ranking represents the Web's most popular websites, per Alexa traffic estimates. We requested the global Top 100 list as well as country-specific Top 100 lists. The country lists are based on the countries that the websites are popular in, not based on where the servers are physically located.

Alexa aggregates browsing history from millions of Alexa toolbar users,[14] which it complements with a website-embedded analytics script.[15] They compute a ranking based on the resulting corrected traffic estimates, which combines counts of unique visitors and page loads from the two data sources. Their traffic ranking combines all subdomains into a single entry for the website "unless [they] are able to automatically identify them as personal home pages or blogs."[14]

We requested these lists on February 13, 2017. Based on how Alexa computes traffic estimates, this reflects popularity over a three-month period prior to that day.[14] Several of the websites were unreachable on the day of testing, so we omitted them from our results.

Google Top 100. The Google Transparency Report provides HTTPS statuses for a list of 100 top websites. It is intended to represent highly popular, *non-Google* websites. The list was created using a mix of public data (including the Alexa ranking) and Google proprietary data, based on browsing habits in early 2016.[11] The lack of Google websites on the list is notable because the Alexa 100 lists include many Google websites.

Alexa Million. The Alexa Top Million represents a broad snapshot of the active Internet. Although the list's official name alludes to popularity, everything after the first 100,000 is considered part of the long tail of the Internet.[14] The list sees substantial churn, and websites are sometimes unreachable after only a few days. Statistics based on the Alexa Top Million should therefore be viewed as reflecting a broad developer experience, rather than reflecting truly "top" sites. We refer to the list as the "Alexa Million" throughout this paper to avoid the impression that all of the websites on the list are popular. We requested the Alexa Million on April 11, 2016, October 21, 2016, and February 3, 2017.

IPv4 hosts. Censys exposes an Internet-wide view of servers. They "use ZMap to perform single-packet host discovery scans against the IPv4 address space" [10]. Once a host is found, they perform a TLS handshake and record the result. Some — perhaps many — of the full set of responding servers are hobbyist machines, defunct websites, home devices, app backends, etc. We queried Censys on February 13, 2017.

4.3 Results

HTTPS support increased from 2016 to 2017. However, each list that we examined yielded a different HTTPS adoption rate (Table 2). Popular websites are more likely to support HTTPS, and support also varies by region.

4.4 Adoption over time

Is HTTPS support increasing? (Yes.)

Top websites. HTTPS support increased dramatically among the Google Top 100 from February 2016 to February 2017 (Figure 6). Over the course of a year, the number of Google Top 100 websites with basic HTTPS support rose from 39% to 54% (15 points). The number of websites with default HTTPS nearly doubled, increasing from 24% to 44% (20 points).

Long tail. We also observed a big increase in HTTPS support among the Alexa Million from April 2016 to February 2017 (Table 3). In less than a full year, the number of Alexa Million websites with basic HTTPS support rose from 30% to 40%. This continues the growth previously observed by Durumeric et al. in 2012-2013 [11]. On the other hand, the number of websites with default HTTPS remained low, increasing from 5% to only 10%.

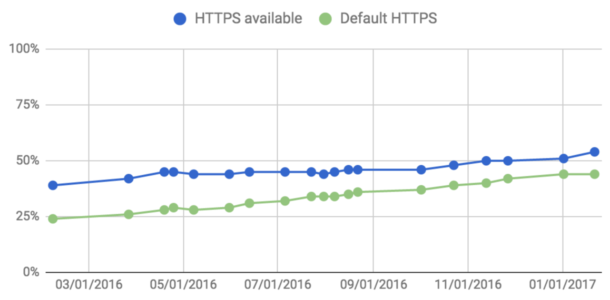

Figure 6: HTTPS support among the Google Transparency Report Top 100, recorded over a year.

	4/2016	10/2016	2/2017
HTTPS available	30%	34%	40%
Default HTTPS	5%	8%	10%
HSTS	1%	3%	3%
HSTS preloading	0.2%	0.2%	0.3%
HPKP	0.4%	0.5%	0.5%

Table 3: Results of testing the Alexa Million with the Mozilla Observatory on three dates.

4.5 Website popularity

Are popular websites more likely to be HTTPS? (Yes.)

As Table 2 shows, the percentage of websites that support HTTPS depends on the list. Websites on the "top" lists are more likely to support HTTPS than others.

Top websites. A majority of all three top website lists (HTTPSWatch Global, Google Top 100, and Alexa Top 100 Global) support HTTPS. Of those three, the Alexa Top 100 Global has the highest rate of HTTPS availability. However, the Alexa Top 100 is not a very diverse list: twenty of the websites are owned by Google, six are owned by Microsoft, and three are owned by Amazon. The Google websites all support HTTPS but not by default (due to an initial HTTP redirection from `google.com` to `www.google.com`). The Google Top 100 avoids this skew by removing Google's own websites; it accordingly yields a lower HTTPS availability rate but a higher default HTTPS rate.

Long tail. We see a steady decrease in HTTPS support as websites get less popular. Of the Alexa Top 100, 87% support HTTPS; of the top 1000, 70% support HTTPS; of the top 10,000, 60% support HTTPS; of the top 100,000, 51% support HTTPS. The full Alexa Million represents the long-but-active tail of the Web, and only 40% support HTTPS (10% by default). IPv4 hosts — representing the long tail and flotsam of the Internet — are even less likely to support HTTPS (10%).

	HTTPS available	Default HTTPS
Brazil	65%	15%
Canada	77%	21%
France	67%	16%
Germany	86%	27%
India	68%	16%
Indonesia	71%	13%
Japan	57%	19%
Mexico	80%	19%
Russia	80%	24%
South Korea	75%	14%
Spain	75%	21%
Turkey	73%	17%
United States	81%	18%
Global	*87%*	*23%*

Table 4: HTTPS support rates among the Alexa Top 100 for each country in February 2017, as tested with the Mozilla Observatory on February 13, 2017.

4.6 Regional disparities

Is HTTPS support equal worldwide? (No.)

We observe different rates of HTTPS support among the Alexa Top 100 lists for fourteen countries (Table 4).

High HTTPS support. Websites that are popular in Germany, the United States, Mexico, and Russia are the most likely to support HTTPS (80% or greater).

Low HTTPS support. Websites that are popular in Japan are the least likely to support HTTPS (57%). To our surprise, Brazil, France, and India are not far behind (65%, 67%, and 68%).

Comparison to browser metrics. We expected to see a clear relationship between regional HTTPS usage and server support. For example, we expected that Japan and South Korea would have very low HTTPS support rates. However, that is not the case: Japanese HTTPS support rates are only slightly lower than others, and South Korea falls in the middle. On the other hand, Indian people have high HTTPS usage rates despite the Indian Top 100 having a relatively low HTTPS support rate.

This discrepancy can be explained by considering that website popularity is not distributed evenly even within the Top 100. People might spend much more time on the first three websites (or first five, or first fifteen...), placing more weight on the HTTPS status of those websites.

Global vs regional. The Alexa Top 100 Global list reports a higher HTTPS support rate (87%) than any individual country list. How can this be? A key insight is that the global list includes a mix of (a) websites that are popular across many countries, and (b) the most popular websites from the largest countries. The websites on the Top 100 list are more popular overall than the websites on the country-specific lists — and the websites at the very top tend to support HTTPS.

One intriguing artifact of the popularity distribution is that the global list has twenty Google websites. This occurs because nineteen large countries each have a popular regional Google variant (e.g., `google.de` and `google.es`), and `google.com` is popular across many countries. As a result, the global list doesn't represent any single person's normal browsing — no one visits all of the different Google variants! In contrast, someone would likely be familiar with most of the websites on their country's Top 100 list.

5 Network measurements

In addition to client and server measurements, we can also observe HTTPS adoption from the network. HTTP and HTTPS operate on different TCP ports (80 vs. 443), so it is easy to identify HTTP and HTTPS traffic in a given packet flow. We can use network measurements to assess how much of the web is being carried over HTTPS across *all* clients and servers being used over a given network. This includes non-browser clients (which are missing from browser telemetry) and less popular sites (which might not be covered by scans).

5.1 MAWI sample point F

We derive our network observations from the public domain MAWI data set published by the WIDE project,[16] specifically from their sample point F [23]. This data set includes one 15-minute snapshot of Internet traffic per day, taken at a connection point between the WIDE backbone network and a transit provider. Between 160GB and 590GB of HTTP and HTTPS traffic passes this observation point during each collection window.

We report the network data in terms of bytes and packets. Top-level page loads, subresources, and non-Web traffic are all grouped together in this data set.

5.2 Results

Figure 7 shows the fraction of bytes and packets that were sent over HTTPS, from 2014 to 2017.

Traffic over time. By both the byte and packet metrics, HTTPS experienced significant growth from January 2014 to January 2017. The percentage of network traffic using HTTPS grew from around 20% of web traffic to around 40%. In addition, the byte and packet ratios converged, which suggests that HTTPS is increasingly being used for the same types of activities as HTTP.

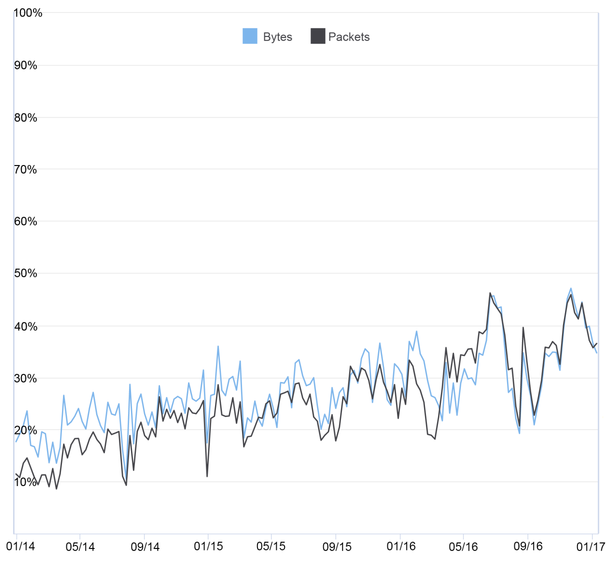

Figure 7: HTTPS as a percentage of all network traffic (HTTP+HTTPS) at the MAWI sample point F, weekly from January 2014 to January 2017.

Comparison to browser metrics. The HTTPS rate as seen by this link is lower than the HTTPS usage rates observed by browser telemetry. We hypothesize that this might be due to one (or more) of three factors:

- These observations are from a network in Japan, where browser telemetry shows low HTTPS usage.
- Non-browser HTTP clients might have a lower rate of HTTPS usage. They are covered by this measurement and not browser telemetry.
- Content served over HTTP could be different than content served over HTTPS in a way that skews measurements by byte or packet (e.g., streaming video might be more common over HTTP).

The disparity between volume-oriented and connection- or pageload-oriented metrics is similar to an earlier observations made in a residential ISP in 2014 [29]. That study observed the prevalence of HTTPS in terms of both traffic volumes (in bytes) and connections, the latter being similar to the Firefox transaction metric. (Technically a lower bound, since multiple transactions can be carried on a single connection.) As of 2014, they observed that HTTPS was 13.8% of download traffic by volume, but 44.3% of connections, both trending upward over time.

6 Implications

We discuss the implications of our findings for HTTPS outreach, discussion, and policies.

6.1 HTTPS adoption growth

We have seen tremendous growth in HTTPS adoption, from four perspectives:

- **Top websites.** Default HTTPS support among the Google Top 100 nearly doubled in 2016, rising from a quarter to nearly half (Figure 6). We attribute this to growing public demand for HTTPS and a desire among top websites to use new HTTPS-only features like Service Workers.
- **Long tail websites.** Among the Alexa Million, basic HTTPS availability grew from 30% to 40% over 2016, and support for HTTPS by default doubled to 17% (Table 3). We attribute the growth in the long tail to Let's Encrypt and publishing platforms that now support HTTPS (such as Squarespace).
- **End user perspective.** A majority of desktop browsing now occurs over HTTPS. HTTP is still dominant on Android by some measures, but HTTPS usage on Android is growing and poised to soon become the majority by all measures. Figure 1 shows the substantial growth in HTTPS usage (per extended page loads) from July 2014 to February 2017 in Chrome. The growth in HTTPS usage is a direct consequence of the growth in server support.
- **Network traffic.** HTTPS traffic doubled as a percentage of all web traffic, by byte and by packet, from 2014 to 2017 (Figure 7).

We view the steady growth as a sign that HTTPS promotion efforts are succeeding.

6.2 Choice of metric

We would have liked to find a single metric that captures HTTPS adoption. Such a metric would be useful for making HTTPS-related decisions and tracking the impact of adoption efforts. Unfortunately, this unified metric is elusive. The browser metrics that we investigated each have their own nuances (Section 3.2), each yielding different HTTPS usage rates (Figure 3, Table 1). Similarly, server support rates depend on the set of websites and level of HTTPS support desired (Table 2). Network-based metrics have broader scope but less detail than browser metrics (Section 5).

We recommend choosing between metrics based on the following guidelines:

- Use the Alexa Million to measure developer impact.
- Use page load metrics to measure the user impact of browser or ecosystem changes that affect page loads. The strict page load metric is preferable because it is more conservative.
- Use a time-based metric to measure the user impact of changes to long-lasting browser UI changes.
- Consider different demographics. Split user metrics by OS and country, and split Alexa sites by country.

In this section, we give examples of applying those recommendations to scenarios that we have encountered.

6.2.1 Requiring HTTPS for APIs

Mozilla Firefox and Google Chrome have begun requiring HTTPS for some developer APIs like Service Workers [1] and the geolocation API [24]. Their goal is to prevent powerful browser APIs from being abused by network attackers. Browser vendors and standards bodies may wish to know what the compatibility cost is when discussing whether to require HTTPS for a new API.

Developer frustration and low feature adoption rates are the two main concerns in this scenario. How many websites will be able to use the feature if its usage is restricted to HTTPS pages? We recommend measuring how many of the Alexa Million support HTTPS by default, since that list represents a wide set of actively visited websites. The number is currently low, which suggests that many developers will need to do additional work to use HTTPS-only features. This should not necessarily deter browser vendors from setting those restrictions, but understanding the ecosystem status can help set expectations for the reaction to a HTTPS restriction.

6.2.2 Changing security indicators

Browser security indicators have slowly changed over time. Each iteration is accompanied by concerns about how the change affects people's perceptions of security states (e.g., [16]). For example, people might become desensitized to a negative or frightening security indicator for HTTP if they see it all the time.

If the proposed UI treatment is displayed once at page load, we recommend using a page load-based metric to evaluate how often it will be shown. If the proposed UI treatment would be permanently associated with the loaded page, then we instead recommend using a time-based metric. A time-based metric is appropriate because it measures how long people actually see the new browsing treatment. We do not think it is necessary to consult server support rates when measuring the impact of UI changes, since UI is between the browser and end user.

When evaluating UI changes, we recommend splitting the relevant HTTPS metrics by operating system and country to see whether any user populations will be disproportionately impacted. In particular, it would be worth understanding how the change will impact East Asian countries due to their lower HTTPS usage.

6.2.3 Handling unknown protocols

If someone types a new website `www.example.com` into a URL bar, should the browser load the website over HTTP or HTTPS? All major browsers currently default to HTTP in this situation, unless a preloaded HSTS header instructs otherwise. At some point, browsers should change their behavior to default to HTTPS instead. This change cannot be made lightly because it carries risk for websites that are still unavailable or appear broken (e.g., due to mixed content) over HTTPS.

When considering such a change, one might be concerned about end user pain (seeing broken pages). We recommend measuring the impact of a proposal in terms of strict page loads. The strict page load metric is a suitable measure of user pain in this situation because the disappointing event might occur at page load, and it is more conservative than the extended page load metric.

If the proposal results in slowing, breaking, or blocking some websites, one might also be concerned about developer frustration. We therefore also recommend measuring the impact of a proposal on developers in terms of the Alexa Million, to gain insight into the long-yet-still-active tail of the Web.

6.3 Impact of top websites

Our data suggests that top websites drive client HTTPS usage on desktop, but Android is more sensitive to server support among the long tail. We observe:

- Server support for HTTPS is more common for top websites than for the long tail (the Alexa Million or IPv4 hosts). The difference is especially pronounced when considering how many support HTTPS by default (Table 2).

- Websites that use the History API and fragment navigation comprise a large amount of HTTPS web browsing. We see this in Figure 3 by comparing the extended page load metric (which includes those types of navigations) to the strict page load metric (which doesn't). The extended page load metric is 20 points higher than the strict page load metric on desktop Chrome, and 9 points higher on Android Chrome. These navigations are associated with highly engineered, dynamic websites.

- Compared to desktop clients, Android clients load fewer HTTPS pages and spend less time on HTTPS pages in Chrome (Figure 3). We attribute this primarily to the popularity of apps on Android, notably including the Google Search and Facebook apps. Long tail traffic remains in the browser.

- Not only do people spend more time on HTTPS websites overall on desktop Chrome, but the websites that people spend a very long time looking at are more likely to be HTTPS. The distributions for HTTP pages and HTTPS pages are similar log-normal distributions, differing only in the very long tail: 1% of HTTP page loads are kept open for more than an hour, whereas 2% of HTTPS page loads are kept open for more than an hour. An example of this might be keeping GMail or Facebook open all day, frequently returning to the tab.

We conclude that a small number of dynamic websites account for a disproportionate percentage of desktop web browsing. As these websites move to HTTPS by default, HTTPS usage on desktop grows. The effect is smaller on Android than desktop, where people spend more time browsing HTTP websites in the long tail.

6.4 Benchmarking HTTPS adoption

Presenters or journalists may want to provide a summary statistic when discussing the current state of HTTPS. In such a situation, we recommend using the following:

- To give an overview of the end user experience, use Chrome's time-in-foreground metric (split by OS). The time-based metric is intuitive and not sensitive to tabbed browsing habits.

- To illustrate trends among influential websites, scan the Google Top 100. The Google Top 100 list has less repetition than the Alexa Top 100, but it's still grounded in user data (unlike HTTPSWatch).

- To track progress among the long tail, scan the Alexa Million. It has a diverse mix of websites, 90% of which are considered part of the long tail but all of which have been visited by real people over the last three months.

6.5 HTTPS by default

Significant work remains to continue shifting the ecosystem. The long tail (Alexa Million) and very long tail (IPv4) still have little HTTPS support, and support among top websites has only recently come close to 50%.

At current rates, we predict:

- Top websites will be almost entirely HTTPS within a year and a half. Half have moved, more are preparing to move, and the reminder will feel pressured to meet the changing industry standard.

- Widespread HTTPS adoption among the long tail will take five more years unless a tool, hosting service, or outreach effort yields a breakthrough.

- HTTPS usage on desktop will be largely HTTPS within two years, due to an emphasis on top websites. On Android, it will depend on trends in the app vs web ecosystem.

6.6 Future outreach

We identified several areas where additional HTTPS outreach could yield benefits:

- East Asia lags behind the rest of the world in HTTPS adoption. Understanding Japanese and South Korean developers' and users' concerns (or lack of interest) could help address this.

- Moving the long tail to HTTPS should help increase HTTPS usage on Android, which currently lags behind desktop. This is more challenging than doing outreach to top websites because the outreach will need to have massive scale. Moving this long tail will require the change at points of centralization that can upgrade many sites at once, e.g., hosting providers or server software vendors.

- We should encourage new top websites to enable HTTPS. Much of the progress in 2016 came from top websites with *some* HTTPS support transitioning to HTTPS by default (Figure 6); as a result, most of the top websites now have either default HTTPS or no HTTPS at all.

7 Repeatability

Our metrics can be tracked or repeated by others. Google releases summary statistics from Chrome HTTPS telemetry weekly as part of the Google Transparency Report[17]. Mozilla publishes aggregate Firefox telemetry on their telemetry website. [18]

Our server scans can be repeated by others using the information in Section 4. Mozilla provides access to the Mozilla Observatory through a public API.[9] One exception is our scan of the Google Top 100, which used the proprietary Googlebot. However, we provide the results of those scans weekly as part of the Google Transparency Report.[19] Alternately, scans of the Google Top 100 can be run by anyone using the Mozilla Observatory.

8 Acknowledgments

We thank Emily Schechter, Tim Willis, Rutledge Chin Feman, and Hubert Chao for their work on the Google Transparency Report; Bryan McQuade and Vern Paxson for their insightful discussion; and David Adrian and Zakir Durumeric for answering questions about Censys.

A Archived list contents

Where possible, we provide copies of the lists that we scanned in Section 4.2. Please be aware that the lists contain websites with adult material.

A.1 Alexa

Amazon provides archived copies of Alexa's rankings via the Alexa Web Services API: `https://aws.amazon.com/alexa/`.

A.2 HTTPSWatch

www.baidu.com, www.bing.com, duckduckgo.com, www.google.com, www.sohu.com, www.yandex.ru, www.yahoo.com, www.linkedin.com, www.facebook.com, www.twitter.com, www.pinterest.com, instagram.com, www.reddit.com, www.youtube.com, vine.co, www.match.com, www.okcupid.com, disqus.com, store.apple.com, www.amazon.com, www.bestbuy.com, www.ebay.com, www.craigslist.org, www.target.com, www.walmart.com, www.cvs.com, www.homedepot.com, www.barnesandnoble.com, www.box.com, www.dropbox.com, drive.google.com, www.icloud.com, onedrive.live.com, www.tarsnap.com, www.blogger.com, medium.com, squarespace.com, staff.tumblr.com, wordpress.com

A.3 Google Transparency Report

aliexpress.com, amazon.co.jp, amazon.co.uk, amazon.com, amazon.de, bongacams.com, chaturbate.com, cnet.com, facebook.com, instagram.com, linkedin.com, mail.ru, netflix.com, nih.gov, nytimes.com, ok.ru, paypal.com, pinterest.com, reddit.com, seznam.cz, softonic.com, taobao.com, theguardian.com, tmall.com, tripadvisor.com, tumblr.com, twitter.com, vk.com, whatsapp.com, wikimedia.org, wikipedia.org, wordpress.com, xhamster.com, yahoo.com, yandex.ru, yelp.com, amazon.in, apple.com, baidu.com, beeg.com, imgur.com, sohu.com, stackoverflow.com, t.co, wp.pl, xvideos.com, 360.cn, alibaba.com, amazonaws.com, ask.com, ask.fm, bbc.co.uk, bing.com, chinadaily.com.cn, cnn.com, craigslist.org, dailymail.co.uk, dailymotion.com, daum.net, ebay.co.uk, ebay.com, fc2.com, forbes.com, globo.com, gmw.cn, go.com, goal.com, goo.ne.jp, hao123.com, hausou.com, imagebam.com, imdb.com, live.com, microsoft.com, milliyet.com.tr, mirror.co.uk, msn.com, naver.com, office.com, olx.biz.id, onet.pl, pornhub.com, pzy.be, qq.com, rakuten.co.jp, redtube.com, sina.com.cn, soso.com, telegraph.co.uk, tianya.cn, uol.com.br, weibo.com, wikia.com, wikihow.com, xinhuanet.com, xnxx.com, yahoo.co.jp, youporn.com

References

[1] Using Service Workers. `https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API/Using_Service_Workers`.

[2] Introducing the Security Panel in DevTools, January 2016. Chromium Blog.

[3] Communicating the Dangers of Non-Secure HTTP, January 2017. Mozilla Security Blog.

[4] AAS, J. Let's Encrypt 2016 In Review, January 2017. Let's Encrypt Blog.

[5] AKHAWE, D., AMANN, B., VALLENTIN, M., AND SOMMER, R. Here's My Cert, So Trust Me, Maybe? Understanding TLS Errors on the Web. In *World Wide Web Conference* (2013).

[6] AKHAWE, D., AND FELT, A. P. Here's My Cert, So Trust Me, Maybe? Understanding TLS Errors on the Web. In *USENIX Security Symposium* (2013).

[7] BAHAJJI, Z. A., AND ILLYES, G. HTTPS as a ranking signal, August 2014. `https://webmasters.googleblog.com/2014/08/https-as-ranking-signal.html`.

[8] COOPER, D., SANTESSON, S., FARRELL, S., BOEYEN, S., HOUSLEY, R., AND POLK, W. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, Internet Engineering Task Force, May 2008.

[9] DIERKS, T., AND RESCORLA, E. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, Internet Engineering Task Force, Aug. 2008.

[10] DURUMERIC, Z., ADRIAN, D., MIRIAN, A., BAILEY, M., AND HALDERMAN, J. A. A Search Engine Backed by Internet-Wide Scanning. In *22nd ACM Conference on Computer and Communications Security* (2015).

[11] DURUMERIC, Z., KASTEN, J., BAILEY, M., AND HALDERMAN, J. A. Analysis of the https certificate ecosystem. In *Proceedings of the 2013 Conference on Internet Measurement Conference* (New York, NY, USA, 2013), IMC '13, ACM, pp. 291–304.

[12] DURUMERIC, Z., WUSTROW, E., AND HALDERMAN, J. A. 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., 2013), USENIX, pp. 605–620.

[13] EASTLAKE, D. Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066, Internet Engineering Task Force, Jan. 2011.

[14] FAHL, S., ACAR, Y., PERL, H., AND SMITH, M. Why eve and mallory (also) love webmasters: a study on the root causes of SSL misconfigurations. In *ACM Symposium on Information, Computer and Communications Security* (2014).

[15] FELT, A. P., AINSLIE, A., REEDER, R. W., CONSOLVO, S., THYAGARAJA, S., BETTES, A., HARRIS, H., AND GRIMES, J. Here's My Cert, So Trust Me, Maybe? Understanding TLS Errors on the Web. In *Conference on Human Factors and Computing Systems* (2015).

[16] FELT, A. P., REEDER, R. W., AINSLIE, A., HARRIS, H., WALKER, M., THOMPSON, C., ACER, M. E., MORANT, E., AND CONSOLVO, S. Rethinking connection security indicators. In *Symposium on Usable Privacy and Security* (2016).

[17] FEMAN, R. C., AND WILLIS, T. Securing the web, together, March 2016. Google Security Blog.

[18] FIELDING, R., GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P., AND BERNERS-LEE, T. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, Internet Engineering Task Force, June 1999.

[19] HODGES, J., JACKSON, C., AND BARTH, A. HTTP Strict Transport Security (HSTS). RFC 6797, Internet Engineering Task Force, Nov. 2012.

[20] HOFFMAN-ANDREWS, J. Verizon Injecting Perma-Cookies to Track Mobile Customers, Bypassing Privacy Controls, November 2014. Electronic Frontier Foundation.

[21] HOLZ, R., BRAUN, L., KAMMENHUBER, N., AND CARLE, G. The ssl landscape: A thorough analysis of the x.509 pki using active and passive measurements. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference* (New York, NY, USA, 2011), IMC '11, ACM, pp. 427–444.

[22] HUSÁK, M., ČERMÁK, M., JIRSÍK, T., AND ČELEDA, P. Https traffic analysis and client identification using passive ssl/tls fingerprinting. *EURASIP J. Inf. Secur. 2016*, 1 (Dec. 2016), 30:1–30:14.

[23] KATO, M., CHO, K., HONDA, M., AND TOKUDA, H. Monitoring the Dynamics of Network Traffic by Recursive Multidimensional Aggregation. In *OSDI* (2012).

[24] KINLAN, P. Geolocation API Removed from Unsecured Origins in Chrome 50, April 2016. `https://developers.google.com/web/updates/2016/04/geolocation-on-secure-contexts-only`.

[25] KONIGSBURG, E., AND WAN, V. HTTPS on NYTimes.com, January 2017. NYTimes: OPEN: All the code that's fit to printf.

[26] KRAVETS, D. Meet the tech company performing ad injections for Big Cable, September 2014. Ars Technica.

[27] LEVILLAIN, O. *A study of the TLS ecosystem*. Theses, Institut National des Télécommunications, Sept. 2016.

[28] MARCZAK, B., WEAVER, N., DALEK, J., ENSAFI, R., FIFIELD, D., MCKUNE, S., REY, A., SCOTT-RAILTON, J., DEIBERT, R., AND PAXSON, V. China's Great Cannon, April 2015. The Citizen Lab.

[29] NAYLOR, D., FINAMORE, A., LEONTIADIS, I., GRUNENBERGER, Y., MELLIA, M., MUNAFÒ, M., PAPAGIANNAKI, K., AND STEENKISTE, P. The cost of the "s" in https. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies* (New York, NY, USA, 2014), CoNEXT '14, ACM, pp. 133–140.

[30] NOTTINGHAM, M. Securing the Web, January 2015. W3C.

[31] RESCORLA, E. HTTP Over TLS. RFC 2818, Internet Engineering Task Force, May 2000.

[32] SCHECHTER, E. Moving towards a more secure web, September 2016. Google Security Blog.

[33] SCHILLACE, S. Default https access for Gmail, January 2010. Official Gmail blog.

[34] TWITTER. Securing your Twitter experience with HTTPS, February 2012. Official Twitter blog.

[35] VEDITZ, D., BARTH, A., AND WEST, M. Content security policy level 2. W3C recommendation, W3C, Dec. 2016. https://www.w3.org/TR/2016/REC-CSP2-20161215/.

[36] VELAN, P., ČERMÁK, M., ČELEDA, P., AND DRAŠAR, M. A survey of methods for encrypted traffic classification and analysis. *Netw. 25*, 5 (Sept. 2015), 355–374.

[37] WEST, M. Mixed content. Candidate recommendation, W3C, Aug. 2016. https://www.w3.org/TR/2016/CR-mixed-content-20160802/.

Notes

[1]`https://letsencrypt.org`

[2]`https://https.cio.gov`

[3]`https://www.ssllabs.com/ssltest`

[4]E.g., `https://developers.googleblog.com/2016/11/heres-to-more-https-on-the-web.html`

[5]The Chromium implementation and histogram definition are open source. They are available at `cs.chromium.org`; the histogram is named `Navigation.MainFrameSchemeDifferentPage`. The Firefox implementation and histogram definition are also open source and available at `searchfox.org`.

[6]The Chromium implementation and histogram definition are open source. They are available at cs.chromium.org; the histogram is named `Navigation.MainFrameScheme`.

[7]The Firefox implementation is open source at `searchfox.org`.

[8]`http://www.nielsen.com/us/en/insights/news/2011/infographic-the-most-valuable-digital-consumers.html`

[9]`https://observatory.mozilla.org`

[10]`https://www.google.com/transparencyreport/https/grid/`

[11]`https://www.google.com/transparencyreport/https/faq/`

[12]`https://httpswatch.com/about`

[13]`https://censys.io`

[14]`https://support.alexa.com/hc/en-us/articles/200449744-How-are-Alexa-s-traffic-rankings-determined-`

[15]`https://support.alexa.com/hc/en-us/articles/200450354`

[16]`http://mawi.nezu.wide.ad.jp/mawi/`

[17]`https://www.google.com/transparencyreport/https/metrics/`

[18]`https://telemetry.mozilla.org`

[19]`https://www.google.com/transparencyreport/https/grid/`