Does "www." Mean Better Transport Layer Security?

Eman Salem Alashwali
University of Oxford
Oxford, United Kingdom
King Abdulaziz University (KAU)
Jeddah, Saudi Arabia
eman.alashwali@cs.ox.ac.uk

Pawel Szalachowski
Singapore University of Technology and Design (SUTD)
Singapore, Singapore
pawel@sutd.edu.sg

Andrew Martin
University of Oxford
Oxford, United Kingdom
andrew.martin@cs.ox.ac.uk

Abstract

Experience shows that most researchers and developers tend to treat plain-domains (those that are *not* prefixed with "www" sub-domains, e.g. "example.com") as synonyms for their equivalent www-domains (those that are prefixed with "www" sub-domains, e.g. "www.example.com"). In this paper, we analyse datasets of nearly two million plain-domains against their equivalent www-domains to answer the following question: *Do plain-domains and their equivalent www-domains differ in TLS security configurations and certificates? If so, to what extent?* Our results provide evidence of an interesting phenomenon: plain-domains and their equivalent www-domains differ in TLS security configurations and certificates in a non-trivial number of cases. Furthermore, www-domains tend to have stronger security configurations than their equivalent plain-domains. Interestingly, this phenomenon is more prevalent in the most-visited domains than in randomly-chosen domains. Further analysis of the top domains dataset shows that 53.35% of the plain-domains that show one or more weakness indicators (e.g. expired certificate) that are not shown in their equivalent www-domains perform HTTPS redirection from HTTPS plain-domains to their equivalent HTTPS www-domains. Additionally, 24.71% of these redirections contains plain-text HTTP intermediate URLs. In these cases, users see the final www-domains with strong TLS configurations and certificates, but in fact, the HTTPS request has passed through plain-domains that have less secure TLS configurations and certificates. Clearly, such a set-up introduces a weak link in the security of the overall interaction.

ACM Reference Format:
Eman Salem Alashwali, Pawel Szalachowski, and Andrew Martin. 2019. Does "www." Mean Better Transport Layer Security?. In *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES 2019) (ARES '19), August 26–29, 2019, Canterbury, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3339252.3339277

1 Introduction

The "www." (world-wide web) prefix has become a de facto standard for domains running websites. Plain-domains (those without the "www." prefix, e.g. "example.com") are usually treated as synonyms for their equivalent www-domains (those that are prefixed with "www.", e.g. "www.example.com").

As a concrete example, from an application development perspective, recently, in August 2018, Google's `Chrome`[1] version 69.0.3497.81 decided to hide the "www" and "m" ("m" for mobile) sub-domains from the steady-state URL in `Chrome`'s address bar by default [21]. Google describes these sub-domains as "trivial" in `Chrome`'s custom settings, where users can enable displaying these sub-domains through the following URI: "chrome://flags/#omnibox-ui-hide-steady-state-url-scheme-and-sub-domains". This change by Google has received criticism and media attention from the security community, as shown in [21][26][7]. One of the reasons for not welcoming this change is that it can cause confusion [21]. For example, two addresses, one with a "www" sub-domain and another without a "www" sub-domain can point to completely different websites [21]. One user reported `Chrome`'s new behaviour of hiding the "www" and "m" sub-domains as a bug [16]. In the same report [16], another user has pointed out that the `Safari` mobile browser also hides the "www" sub-domains from its address bar [16]. Subsequently, other users have reported that the Google search engine sometimes hides the "www" sub-domains in search results and uses plain-domains format [1]. However, we are unable to reproduce the issue (bug) reported in [16] regarding `Chrome`'s new behaviour since `Chrome` does not provide an archive for old versions. Nevertheless, the reported bug includes several confirming responses [16], in addition to media reports, such as [21][26][7], which provide sufficient evidence that the issue has existed in that particular version, either in the desktop, or the mobile version, or both. Our test of `Chrome`'s latest version shows that subsequent versions (as of March 2019, version 73.0.3683.86) have reverted this change from a default to a custom setting.

From a research perspective, in particular, on Internet measurement studies which are concerned with examining domains' TLS security configurations and certificates, we observe different treatments for the examined domain names. A considerable number of these studies make use of the `Alexa` top domains list [4]. `Alexa` represents domains as plain-domains, except for a small percentage[2] of domains that include sub-domains including "www" sub-domains. Some studies, such as [19], add the "www" sub-domains to the examined domains, and report the results based on TLS handshakes with www-domains. Other studies, such as [15], first try to establish a TLS handshake with the examined domain "as is", and if the handshake with the domain "as is" has failed, they make a second handshake with the domain's equivalent www-domain, and report the results. The third line of

ARES '19, August 26–29, 2019, Canterbury, United Kingdom
2019. ACM ISBN 978-1-4503-7164-3/19/08...$15.00
https://doi.org/10.1145/3339252.3339277

[1]As a shorthand, we use the term `Chrome` for the rest of the paper to refer to Google's `Chrome`.
[2]This percentage is around 4.99% in our `Alexa` dataset.

Internet measurement studies such as [11][13], do not mention adding any "www" sub-domains to the examined domains, hence, we assume that they treat the list's domains (e.g. the `Alexa` list in these studies) "as is". However, it remains unclear whether plain-domains differ from their equivalent www-domains in terms of TLS configurations and certificates? If one type of domains, e.g. www-domains, provides better TLS security configurations than their equivalent plain-domains, or vice versa, to what extent does taking one approach over another affect the overall results? In particular, regarding the domains' adoption of TLS security configurations, e.g. protocol versions, or the domains' certificates status results.

In this paper, we try to answer the following question: *Do plain-domains and their equivalent www-domains differ in TLS security configurations and certificates? If so, to what extent?*

2 Background

The Transport Layer Security (TLS) protocol [25][24], formerly known as Secure Socket Layer (SSL), is one of the most important and widely used security protocols to date. The latest version of TLS is known as TLS 1.3 [25]. We refer to versions prior to TLS 1.3 (in particular, TLS 1.2, TLS 1.1, and TLS 1.0) as pre-TLS 1.3. TLS operates below application layer protocols, to provide data confidentiality, integrity, and authentication. TLS consists of multiple sub-protocols, including the handshake protocol. In the handshake protocol, both communicating parties authenticate each other, and negotiate security-sensitive parameters, including the protocol version and the ciphersuite, that will be used to secure subsequent messages of the protocol. The ciphersuite is an identifier, represented by a string or a hexadecimal value, which defines the cryptographic algorithms (e.g. symmetric encryption algorithm) and their parameters (e.g. key length) that will be used in subsequent messages of the protocol. In pre-TLS 1.3 versions, the ciphersuite defines the key-exchange, authentication, symmetric encryption, and hash algorithms. However, in TLS 1.3, the key-exchange is separated from the ciphersuites, and it is now negotiated in specific extensions.

The version and ciphersuite choices are intrinsic in determining the security guarantees that the protocol can provide in a particular session. Some ciphersuites provide stronger security guarantees than others. For example, Forward Secrecy (FS) guarantees that a compromise in a server's long-term private-key does not compromise past session keys [20]. Similarly, Authenticated Encryption (AE) provides confidentiality, integrity, and authenticity, simultaneously, which provides stronger resilience against attacks over the MAC-then-encrypt schemes [27][5]. The same applies to the protocol version. Each version of the protocol prevents attacks discovered in previous versions, and provides more security guarantees. For example, TLS 1.3 enforces FS key-exchange algorithms and AE ciphersuites (FS+AE) by design. TLS 1.2 supports AE in addition to FS but both are optional, while TLS 1.1 and TLS 1.0 do not support AE ciphersuites.

As depicted in Figure 1, at the beginning of a new TLS handshake, the client sends a `ClientHello` (`CH`) message to the server. The `CH` contains several security-sensitive parameters, including: first, the client's supported versions. In pre-TLS 1.3, this parameter is sent as a single value v_{max_C}. In TLS 1.3, it is sent as a list $[v_1, ..., v_n]$ in the "supported_versions" extension. The v_{max_C} is still included in TLS 1.3 `CH` for backward compatibility, and its value is set to TLS 1.2. Second, the client's supported ciphersuites is sent as a list $[a_1, ..., a_n]$. Third, the client's supported extensions as a list $[e_1, ..., e_n]$ at the end of the message. In TLS 1.3, the extensions must at least include the "supported_versions", while in TLS 1.2, the extensions are optional. Upon receiving the `CH`, the server selects a single mutually supported version v_S and ciphersuite a_S from the client's offer. The server then responds with a `ServerHello` (`SH`) containing v_S and a_S. The server's selection of v_S and a_S is imposed on the client. That is, the server's decision on the version and ciphersuite is the final one. If the server does not support the client's offered versions or ciphersuites, the server responds with a handshake failure alert.

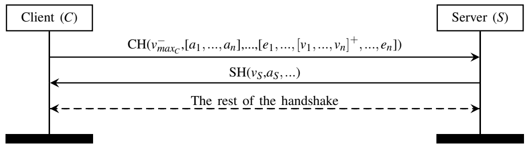
Client (C)
Server (S)
CH($v^{-}_{max_C}$,$[a_1, ..., a_n]$,...,$[e_1, ..., [v_1, ..., v_n]^{+}, ..., e_n]$)
SH(v_S,a_S, ...)
The rest of the handshake

Figure 1: Illustration of the TLS version and ciphersuite negotiation. Parameters followed by "−" are deprecated in TLS 1.3 but still included for backward compatibility, while those followed by "+" are newly introduced in TLS 1.3. The unmarked parameters are mutual to both versions.

3 Dataset

Our study includes two datasets: `top-domains`, which contains 829,873 distinct most-visited domains globally, and `random-domains`, which contains 992,422 distinct random domains. We now define some of the terms that we use throughout the paper: (1) **Main-domains:** domains consisting of a Top Level Domain (TLD) (e.g. "com") prefixed by a single label, and do not have any further sub-domains, e.g. "example.com". (2) **Plain-domains:** domains that are not prefixed with "www" sub-domains, e.g. "example.com". (3) **www-domains:** domains that are prefixed with "www" sub-domains, e.g. "www.example.com". Our scope is limited to generic TLDs (gTLDs), and does not include "multi-level" TLDs such as country-code TLDs (ccTLDs), e.g. "ac.uk". This is to avoid the complexity of distinguishing domains that have sub-domains from domains that have ccTLDs, which is somewhat difficult to achieve with 100% accuracy. In what follows we describe how we build and pre-process each dataset.

3.1 Top-Domains

The `top-domains` dataset is derived from the `Alexa` list of the top one million most-visited domains globally [4], retrieved on December 11, 2018. Since our study targets plain-domains initially, from the one million domains, we extract the domains that are classified as plain-domains and also classified as main-domain (see the previous paragraph for our definitions of plain-domains and main-domains). We end up with 829,873 distinct top domains. To create their equivalent www-domains, we make a replica of the extracted plain-domains, then we prefix each domain with "www.".

3.2 Random-Domains

Our `random-domains` dataset initial size is one million domains, extracted from a large dataset of 54,063,220 domains that have successfully completed TLS handshakes in Amann et al. [6]. The domains in [6] are collected from multiple sources including other lists and previous work. We pre-process our `random-domains` dataset in two steps: First, to maintain consistency with the `top-domains` dataset format, from Amann et al.'s list [6], we extract one million random domains that are classified as main-domains and plain-domains. Second, from the just extracted domains, we exclude the domains that already exist in the `top-domains` dataset, to avoid repetition. We finish with 992,422 distinct random domains. Finally, to create their equivalent www-domains, we create a replica of the dataset, then we prefix each domain with "www.".

4 Methodology

4.1 Data Collection

To collect data, we run a TLS client that takes the domain names from our datasets as input, performs a TLS handshake with each domain, and outputs the handshake's response data. For each dataset, namely the `top-domains` and `random-domains` datasets, the client performs TLS handshakes with plain-domains and their equivalent www-domains concurrently, using two separate TLS client instances. We perform two types of TLS handshakes per domain: one utilising TLS 1.2 client configurations, and the second utilising TLS 1.3 client configurations. Our clients' configurations (mainly, the supported TLS versions and ciphersuites) are based on `Chrome`'s latest version configurations. We choose to base our clients' TLS configurations on `Chrome` because it is the most representative TLS client on the Internet today. As of February 2019, `Chrome`'s usage is 79.7% [28]. To increase our confidence in the obtained results, we run the above-described experiment twice. The first experiment run from the 11th to 12th December 2018, and the second one run from the 5th to 6th March 2019.

The TLS handshakes' responses, which include the servers' TLS versions, ciphersuites, and certificates are first stored in `json` format, parsed, then stored and analysed using the `MySQL` database and queries. In terms of counting the servers' responses, we follow a best-effort approach. That is, we count the responding domains but we do not investigate the reasons for the non-responding ones. However, we eliminate the possibility of servers' rate-limit[3] as a reason for the non-responding domains, as our client performs only two handshakes (TLS 1.2 and TLS 1.3) per domain type (plain-domains and www-domains) per dataset (`top-domains` and `random-domains`). We also eliminate the time gap[4] as a reason for the non-responding domains, as our clients' handshakes with both plain-domains and their equivalent www-domains are performed concurrently. In our analysis, we only consider the domains that responded to both plain-domains and their equivalent www-domains handshakes.

[3]Rate-limit is a technique used to mitigate Denial of Service (DoS) attacks by specifying a threshold for the incoming requests by a particular source address.

[4]By time gap we refer to a situation where a domain can be alive in one handshake (e.g. with plain-domain), but goes down in the second handshake (e.g. with www-domain).

In terms of the TLS client that performs the TLS handshakes, our TLS client is based on `tls-scan` [30], an open source fast TLS scanner capable of performing concurrent TLS connections. We then customise the `tls-scan` to utilise the `OpenSSL 1.1.0g` TLS library for our TLS 1.2 client and the `OpenSSL 1.1.1a` for our TLS 1.3 client. We customise our clients' offered TLS versions and ciphersuites to be equivalent to those offered by `Chrome`'s latest version for our TLS 1.3 client, and `Chrome`'s latest version pre-TLS 1.3 configurations for our TLS 1.2 client. The Server Name Indication (SNI)[5] extension is included by default. We set the concurrency argument to 50 and 25 concurrent connections (these vary between the first and second experiment), and the timeout argument is set to 5 seconds. We run the experiments at the University of Oxford, on computers equipped with 1000 Mbps wired Ethernet cards.

After the TLS scan is complete, to better understand the results, we conduct an HTTPS redirection scan. We build a multi-threaded redirection scanner based on the Python's `Requests` library [23], which provides HTTPS redirection data. The scanner takes as input the domains that have provided responses for both plain-domains and their equivalent www-domains in the TLS 1.3 handshakes in the second TLS scan experiment. The scanner then performs an HTTPS request with each domain without certificate validation, and collects the HTTPS redirection chain from the initial until the final URL (or domain) for each request. We then store and analyse the data using the `MySQL` database and queries. The redirection scan run from the 15th to 17th March 2019 at the University of Oxford. The short time span between the last TLS scan experiment and the redirection scan (around 8 days) allows us to correlate the redirection behaviour to the TLS configurations and certificate behaviour with confidence.

4.2 Ethical Considerations

First, we do not collect any private data. The data we collect are public metadata. Second, we do not perform an exhaustive number of handshakes on any single server. Our client's handshakes can by no means be classified as a Denial of Service (DoS) attack. Third, we use a designated public IPv4 address for the scanning device instead of Network Address Translation (NAT), to avoid potential disturbance to other users in our organisation's network, if a server has blocked our scanning device's IP. Fourth, we use an informative DNS name for our scanning device that contains "TLS probing", to help server administrators identify our device's activity in their logs. Finally, we inform our University's IT and Security teams, so they expect a high volume of outgoing connections from our experiment devices, and to expect some incoming blacklisted certificates from random servers.

4.3 Data Analysis

To answer the first part of our research question (*do plain-domains and their equivalent www-domains differ in TLS security configurations and certificates?*), we count the number of domains that: (a) responded to both plain-domains and their equivalent www-domains TLS handshakes, and (b) provide different values for one or more of the following TLS configurations: (1) TLS

[5]The SNI is a TLS extension that passes the domain name in the TLS handshake in order to get more accurate responses in virtual hosting environments [12].

versions, (2) Ciphersuites, (3) Versions *and* ciphersuites, (4) Versions *or* ciphersuites, (5) Leaf certificate Subject Key Identifiers (SKI), which is an X509 extension that provides the means for identifying certificates with a particular public-key [10], (6) Leaf certificate fingerprints (SHA-1), which provide the means for identifying certificates in general (including all the certificate's fields), (7) IPv4, although not directly a TLS security configuration, but we include it as a useful general result.

To answer the second part of our research question (*If so, to what extent?*), we first define some weakness indicators denoted by *weak*. Then, we analyse plain-domains and their equivalent www-domains against *weak*. In what follows, we list these weakness indicators, and define when they are satisfied by domains:

(1) v.<TLS 1.3: is satisfied by domains that select TLS versions less than TLS 1.3 (the latest version).
(2) v.<TLS 1.2: is satisfied by domains that select TLS versions less than TLS 1.2. Versions less than TLS 1.2 are officially weak and should not be used today.
(3) non-FS: is satisfied by domains that select non-FS ciphersuites (despite the AE or any other properties). We define non-FS ciphersuites[6] as those that do not support the Elliptic-Curve Diffie-Hellman (ECDHE), and are also not negotiated with version TLS 1.3 since TLS 1.3 enforces FS by design in a separate extension. Non-FS ciphersuites provide fewer security guarantees than FS ciphersuites.
(4) non-FS+non-AE: is satisfied by domains that select non-FS+non-AE ciphersuites, i.e. neither provide FS nor AE. We define non-FS+non-AE ciphersuites as those that do not support ECDHE, are not negotiated with version TLS 1.3, do not support the ChaCha20 symmetric encryption, and do not support the GCM symmetric encryption mode. This indicator is not satisfied by domains that select FS+AE, i.e. those that either support ECDHE or negotiated with version TLS 1.3, and support either the ChaCha20 symmetric cipher or the GCM mode. Non-FS+non-AE ciphersuites provide fewer security guarantees than those that provide both FS and AE (FS+AE).
(5) Exp. Certs.: is satisfied by domains that provide expired certificates. The expiration check is performed by the `tls-scan` client against the scan date [30].
(6) Invalid Certs.: is satisfied by domains that provide invalid certificates. The certificate validation is performed by the `tls-scan` client using the `SSL_get_verify_result` function in the `OpenSSL` library against our updated `Ubuntu 18.04` certificates store. It validates the certificate's signature, trust chain, and other verification steps specified in [22].
(7) Key<2048: is satisfied by domains that use RSA certificates with a key length of less than 2048-bit. The minimum recommended RSA key length today is 2048-bit[7].

Then, we count the domains' responses under the following conditions, where *weak* denotes a weakness indicator, *plain* denotes plain-domains that have responses for their equivalent www-domains, and *www* denotes www-domains that have responses for their equivalent plain-domains:

[6]All our definitions are based on `Chrome`'s configurations.
[7]https://www.keylength.com.

Table 1: Responding servers to our TLS clients' handshakes.

Client	Dataset	Size	Type	Responses
`TLS 1.2`	`top-domains`	829,873	plain	691,200 (83.29%)
			www	710,509 (85.62%)
	`random-domains`	992,422	plain	623,869 (62.86%)
			www	620,884 (62.56%)
`TLS 1.3`	`top-domains`	829,873	plain	691,145 (83.28%)
			www	710,496 (85.62%)
	`random-domains`	992,422	plain	622,904 (62.77%)
			www	620,062 (62.48%)

(1) $weak_{plain}$: *weak* is satisfied by *plain*.
(2) $weak_{plain} \wedge \neg weak_{www}$: *weak* is satisfied by *plain* and *weak* is *not* satisfied by *www*.
(3) $weak_{www}$: *weak* is satisfied by *www*.
(4) $weak_{www} \wedge \neg weak_{plain}$: *weak* is satisfied by *www* and *weak* is *not* satisfied by *plain*.

5 Results

As described earlier in the methodology, we have conducted two experiments over a 3-month time span, to increase our confidence in the obtained results. In this section, we only report the results from the latter experiment (March 2019). However, they are consistent with the former experiment (December 2018).

5.1 Responding Servers

Table 1 summarises the number of successful TLS handshake responses from plain-domains and their equivalent www-domains in the `top-domains` and `random-domains` datasets, in both types of client handshakes, TLS 1.2 and TLS 1.3. Responses from top domains are higher than those from random domains. This is mostly due to the fact that the TLS adoption rate in top domains is higher than that in random domains. Furthermore, the `top-domains` dataset is built from a recent `Alexa` list which contains active domains, while the `random-domains` dataset is built from a less recent list from Amann et al. [6], which may contain many inactive domains. We notice that in the `top-domains` dataset, www-domains responses are higher than plain-domains responses by 2.3%. This means that top domains tend to have more active www-domains than plain-domains. On the other hand, in the `random-domains` dataset, both plain-domains and www-domains responses are nearly equal, with slightly more responses from plain-domains than www-domains.

5.2 The Difference is in the Detail

As depicted in Table 2, our results show that the client type, i.e. TLS 1.2 vs. TLS 1.3, does not make a noticeable effect in terms of the exhibited differences in TLS configurations and certificates between plain-domains and their equivalent www-domains. However, in the "version" difference, TLS 1.3 client shows a higher percentages (1.35% and 0.54%) than TLS 1.2 client (0.22% and 0.12%) in the `top-domains` and `random-domains` datasets, respectively. This can be explained by the fact that TLS 1.3 was standardised in 2018, which suggests that domain administrators may have updated the TLS version of one domain, e.g. the plain-domain, but not its equivalent, e.g. the www-domain, or vice versa. In terms of other TLS

Table 2: Differences between plain-domains and their equivalent www-domain against some TLS configurations. The "Responses" column represents the number of domains that responded to *both* plain-domain and their equivalent www-domain TLS handshakes.

Client	Dataset	Responses	Different						
			Version	Ciphersuite	Version ∧ Ciphersuite	Version ∨ Ciphersuite	SKI	Cert.	IPv4
`TLS 1.2`	`top-domains`	678,337	1472 (0.22%)	23,600 (3.48%)	1437 (0.21%)	23,635 (3.48%)	74,433 (10.97%)	75,359 (11.11%)	125,767 (18.54%)
	`random-domains`	614,184	748 (0.12%)	7883 (1.28%)	737 (0.12%)	7894 (1.29%)	52,465 (8.54%)	52,844 (8.60%)	57,037 (9.29%)
`TLS 1.3`	`top-domains`	678,214	9187 (1.35%)	24,819 (3.66%)	9151 (1.35%)	24,855 (3.66%)	74,375 (10.97%)	75,311 (11.10%)	125,861 (18.56%)
	`random-domains`	612,839	3292 (0.54%)	8386 (1.37%)	3282 (0.54%)	8396 (1.37%)	52,318 (8.54%)	52,718 (8.60%)	57,063 (9.31%)

Table 3: Breakdown of some weakness indicators *weak* in plain-domains and their equivalent www-domains based on the TLS 1.3 client handshake responses. The indentation in the "Type/Condition" column indicates that the percentages of the indented row's results are computed over the previous row's results. The percentages of the non-indented row's results are computed over the "Responses" values. Recall: *weak* denotes weakness indicator; $weak_{plain}$ denotes *weak* is satisfied by *plain*; $weak_{plain} \wedge \neg weak_{www}$ denotes *weak* is satisfied by *plain* and *weak* is not satisfied by *www*; $weak_{www}$ denotes *weak* is satisfied by *www*; and $weak_{www} \wedge \neg weak_{plain}$ denotes *weak* is satisfied by *www* and *weak* is not satisfied by *plain*.

Dataset	Responses	Type/Condition	Weakness Indicator (*weak*)						
			v.<TLS 1.3	v.<TLS 1.2	non-FS	non-FS+non-AE	Exp. Cert.	Invalid Cert.	Key<2048
`top-domains`	678,214	$weak_{plain}$	542,267 (79.96%)	15,731 (2.32%)	27,777 (4.10%)	17,368 (2.56%)	24,521 (3.62%)	55,932 (8.25%)	6306 (0.93%)
		$weak_{plain} \wedge \neg weak_{www}$	5280 (0.97%)	1179 (7.49%)	3151 (11.34%)	1907 (10.98%)	2510 (10.24%)	5281 (9.44%)	278 (4.41%)
		$weak_{www}$	539,543 (79.55%)	14,846 (2.19%)	25,184 (3.71%)	15,710 (2.32%)	23,214 (3.42%)	52,773 (7.78%)	6116 (0.90%)
		$weak_{www} \wedge \neg weak_{plain}$	2556 (0.47%)	294 (1.98%)	558 (2.22%)	302 (1.92%)	1203 (5.18%)	2122 (4.02%)	135 (2.21%)
`random-domains`	612,839	$weak_{plain}$	545,924 (89.08%)	27,774 (4.53%)	38,653 (6.31%)	30,621 (5.00%)	55,734 (9.09%)	107,521 (17.54%)	18,781 (3.06%)
		$weak_{plain} \wedge \neg weak_{www}$	1667 (0.31%)	536 (1.93%)	770 (1.99%)	496 (1.62%)	911 (1.63%)	1684 (1.57%)	191 (1.02%)
		$weak_{www}$	545,171 (88.96%)	27,448 (4.48%)	38,152 (6.23%)	30,280 (4.94%)	55,350 (9.03%)	106,663 (17.40%)	18,628 (3.04%)
		$weak_{www} \wedge \neg weak_{plain}$	914 (0.17)%	210 (0.77%)	269 (0.71%)	209 (0.69%)	527 (0.95%)	826 (0.77%)	46 (0.25%)

configurations and certificate differences at both clients TLS 1.3 and TLS 1.2, in general, top domains show higher percentages of differences in the TLS configurations and certificates between plain-domains and their equivalent www-domains than random domains. In general, the most significant differences are in the leaf certificate fingerprints ("Cert."), certificate "SKI", and in the selected "version or ciphersuite". For example, as illustrated in Table 2, in the `top-domains` dataset, more than 11% of plain-domains provide a different certificate fingerprint than their equivalent www-domains. Over 10% of plain-domains provide a different SKI than their equivalent www-domains. Over 3.4% select a different version or ciphersuite than their equivalent www-domains. On the other hand, the `random-domains` dataset shows lower percentages of differences (see Table 2).

5.3 When "`www.`" Means Better TLS Security

For further analysis, we compare plain-domains and their equivalent www-domains against several defined weakness indicators denoted by *weak* (see section 4.3 for further details about the methodology). In this section, we limit our analysis to TLS 1.3 client handshake results as TLS 1.3 handshake is more representative of an updated TLS client today, such as web browsers (recall that in our TLS scan experiments, we perform two types of handshakes for each domain, one utilising TLS 1.3 configurations, and the second utilising TLS 1.2 configurations). As shown in Table 3, it is always the case that there are fewer www-domains that satisfy weakness indicators compared to plain-domains that do so (see Table 3, under the "Type/Condition" column, compare the results of row "$weak_{plain}$" to those of "$weak_{www}$" in each dataset). Also, it is always the case that the number of plain-domains that satisfy a weakness indicator while their equivalent www-domains do not, is higher than the number of www-domains that satisfy a weakness indicator while their equivalent plain-domains do not (see Table 3, under the "Type/Condition" column, compare the results of row "$weak_{plain} \wedge \neg weak_{www}$" to those of "$weak_{www} \wedge \neg weak_{plain}$" in each dataset). For example, as depicted in Table 3, in the `top-domains` dataset, of the 3.62% plain-domains that provide expired certificates, there are 10.24% that provide non-expired certificates by their equivalent www-domains. However, in the same dataset (`top-domains`), of the 3.42% www-domains that provide expired certificates, there are only 5.18% that provide non-expired certificates by their equivalent plain-domains. The same trend appears in all of the weakness indicators we study (see Table 3), which suggests that www-domains tend to have better TLS security configurations and certificates than their equivalent plain-domains.

5.4 Relationship to HTTPS Redirection

To better understand the reasons behind the observed phenomenon, we analyse the HTTPS redirection behaviour of those domains. We input 678,214 top domains and 612,839 random domains that have provided responses for both plain-domains and their equivalent www-domains in the TLS 1.3 latest scan experiment. As depicted in Table 4, from the `top-domains` dataset, 97.55% of plain-domains and 97.68% of www-domains responded to our redirection scan. From the `random-domains` dataset, 95.92% of plain-domains, and 96.01% of www-domains have responded to our redirection scan. Of the responding domains to our redirection scan, in the `top-domains` dataset, 34.14% of HTTPS plain-domains redirected to (i.e. land on) their equivalent HTTPS www-domains, while 23.54% of HTTPS www-domains redirected to their equivalent HTTPS plain-domains. In the `random-domains` dataset, 13.35% of HTTPS plain-domains redirected to their equivalent HTTPS www-domains, while 10.13% of HTTPS

Table 4: Summary of the HTTPS redirection scan results.

Dataset	Size	Type	Responses	Redirection
`top-domains`	678,214	plain	661,596 (97.55%)	225,839 (34.14%)
		www	662,474 (97.68%)	155,921 (23.54%)
`random-domains`	612,839	plain	587,850 (95.92%)	78,449 (13.35%)
		www	588,380 (96.01%)	59,611 (10.13%)

www-domains redirected to their equivalent HTTPS plain-domains. From these results, we conclude that plain-domains tend to redirect to their equivalent www-domains more than www-domains that redirect to their equivalent plain-domains, and HTTPS redirection from plain-domains to their equivalent www-domains is more utilised in `top-domains` than in `random-domains`.

We conduct further analysis on the set of domains that showed at least one weakness indicator in plain-domains, but not in their equivalent www-domains in the TLS scan (see Table 3, domains in row "$weak_{plain} \land \neg weak_{www}$" in both datasets). In the `top-domains` dataset, out of 11,893 top domains that fall into this set and responded to our plain-domains redirection scan, 6345 (53.35%) HTTPS plain-domains redirected to (land on) their equivalent HTTPS www-domains. Of those, 1568 (24.71%) have one or more HTTP (plain-text, unencrypted and unauthenticated) intermediate URLs in the redirection chain. In the `random-domains` dataset, out of the 3369 random domains that fall into that set and responded to our redirection scan, 664 (19.71%) HTTPS plain-domains redirected to their equivalent HTTPS www-domains. Of those, there are 252 (37.95%) HTTP intermediate URLs. The redirection from HTTPS plain-domains to HTTPS www-domains in these domains that show one or more weakness indicators in plain-domains, but not in www-domains, is higher than the overall redirection rate that is presented in Table 4 earlier. Apparently, in this set of domains, domain administrators pay more attention to www-domains security as HTTPS plain-domains are redirected to their equivalent HTTPS www-domains. However, secure redirection should maintain secure TLS configurations and certificates at every point in the redirection chain.

6 Related Work

Chang et al. [9] conducted a study that assessed the HTTPS redirection in top domains. They found that the majority (83.3%) of HTTPS redirections are not secure. They examine application level properties such as the `Strict-Transport-Security` (HSTS) policy adoption. Our work looks at the problem from a different layer. That is, we consider transport layer security configurations and certificates, which have not been considered previously. Wählisch et al. [29] noticed differences in the IP address prefixes between plain-domains and www-domains as part of their empirical analysis of the Resource Public Key Infrastructure (RPKI) deployment on the `Alexa` list of the top one million domains. They pointed out that the differences in the top 100K domains are higher than those in the rest of the domains. Our work examines two different datasets, `top-domains`, and `random-domains`. Additionally, it compares the TLS configurations and certificates, which have not been explored in previous work. Finally, there are several studies that evaluate the cryptographic strengths of TLS servers such as Lee et al. [18], Holz et al. [14], Alashwali [2], and more recently, Kotzias et al. [17], Calzavara et al. [8], and Alashwali et al. [3]. However, none of them have attempted to compare the TLS configurations when connecting to plain-domains and their equivalent www-domains.

7 Conclusion

In this paper, we presented the results of an experiment that aims to explore whether plain-domains and their equivalent www-domains differ in TLS security configurations and certificates? and if so, to what extent? Our results inform the Internet measurement-based research community, domains (servers) administrators, developers, and users alike. First, we provided evidence that there is a difference between plain-domains and their equivalent www-domains in terms of TLS security configurations and certificates in a non-trivial number of cases. The difference is more notable in top domains than in random domains. Second, by defining some weakness indicators and examining when plain-domains and their equivalent www-domains satisfy these indicators, we showed that www-domains tend to have better TLS security than their equivalent plain-domains. Third, we showed that HTTPS redirection from plain-domains to their equivalent www-domains is widely utilised, especially in the top domains, and more significantly in the set of domains that show one or more weakness indicator in plain-domains but not in their equivalent www-domains ($weak_{plain} \land \neg weak_{www}$). In the latter case, users see the final www-domains (URL) with strong TLS security configurations and certificates, but in fact, the HTTPS request has actually passed through plain-domains which have less secure TLS configurations and certificates at previous points in the redirection chain. Even worse, many intermediate URLs in these redirection chains are plain-text HTTP. This introduces a weak link in the system, which may be dangerous for users.

Acknowledgments

We thank the CS's IT and OxCERT teams at the University of Oxford, and the `tls-scan` developer, Binu Ramakrishnan, for technical support. Pawel's work was supported by the SUTD SRG ISTD 2017 128 grant.

References

[1] Lawrence Abrams. 2018. Google Testing Removal of WWW Subdomain from Search Results. https://www.bleepingcomputer.com/news/google/google-testing-removal-of-www-subdomain-from-search-results/ Accessed Dec. 17, 2018.

[2] Eman Salem Alashwali. 2013. Cryptographic Vulnerabilities in Real-Life Web Servers. In *Proceedings of Int. Conference on Communications and Information Technology (ICCIT)*. 6–11.

[3] Eman Salem Alashwali, Pawel Szalachowski, and Andrew Martin. 2019. Towards Forward Secure Internet Traffic. In *Proceedings of Security and Privacy in Communication Networks (SecureComm)*.

[4] Alexa Internet Inc. 2018. Alexa Top Sites. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip Accessed Dec. 11, 2018.

[5] Nadhem J AlFardan and Kenneth G Paterson. 2013. Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. In *Proceedings of Security and Privacy (SP)*. 526–540.

[6] Johanna Amann, Oliver Gasser, Quirin Scheitle, Lexi Brent, Georg Carle, and Ralph Holz. 2017. Mission Accomplished?: HTTPS Security After DigiNotar. In *Proceedings of Internet Measurement Conference (IMC)*. 325–340.

[7] Shawn Brink. 2018. Hide or Show Scheme and WWW Subdomains of URLs in Google Chrome Address Bar. https://www.tenforums.com/tutorials/117616-hide-show-www-subdomains-urls-address-bar-google-chrome.html Accessed Dec. 1, 2018.

[8] Stefano Calzavara, Riccardo Focardi, Matus Nemec, Alvise Rabitti, and Marco Squarcina. 2019. Postcards from the Post-HTTP World: Amplification of HTTPS Vulnerabilities in the Web Ecosystem. In *Proceedings of Security and Privacy (SP)*.

[9] Li Chang, Hsu-Chun Hsiao, Wei Jeng, Tiffany Hyun-Jin Kim, and Wei-Hsi Lin. 2017. Security Implications of Redirection Trail in Popular Websites Worldwide. In *Proceedings of World Wide Web (WWW)*. 1491–1500.

[10] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. 2008. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. https://tools.ietf.org/html/rfc5280#page-28 Accessed Dec. 28, 2018.

[11] Zakir Durumeric, James Kasten, Michael Bailey, and J Alex Halderman. 2013. Analysis of the HTTPS Certificate Ecosystem. In *Proceedings of Internet Measurement Conference (IMC)*. 291–304.

[12] D. Eastlake 3rd. 2011. Transport Layer Security (TLS) Extensions: Extension Definitions. https://tools.ietf.org/html/rfc6066#page-6 Accessed Jun. 19, 2019.

[13] Adrienne Porter Felt, Richard Barnes, April King, Chris Palmer, Chris Bentzel, and Parisa Tabriz. 2017. Measuring HTTPS Adoption on the Web. In *Proceedings of USENIX Security Symposium*. 1323–1338.

[14] Ralph Holz, Lothar Braun, Nils Kammenhuber, and Georg Carle. 2011. The SSL Landscape: a Thorough Analysis of the X.509 PKI Using Active and Passive Measurements. In *Proceedings of Internet Measurement Conference (IMC)*. 427–444.

[15] Lin-Shung Huang, Shrikant Adhikarla, Dan Boneh, and Collin Jackson. 2014. An Experimental Study of TLS Forward Secrecy Deployments. 18, 6 (2014), 43–51.

[16] hugues.a...@gmail.com. 2018. Incorrect Transforms When Stripping Subdomains. https://bugs.chromium.org/p/chromium/issues/detail?id=881410 Accessed Dec. 1, 2018.

[17] Platon Kotzias, Abbas Razaghpanah, Johanna Amann, Kenneth G Paterson, Narseo Vallina-Rodriguez, and Juan Caballero. 2018. Coming of Age: A Longitudinal Study of TLS Deployment. In *Proceedings of Internet Measurement Conference (IMC)*. 415–428.

[18] Homin K Lee, Tal Malkin, and Erich Nahum. 2007. Cryptographic Strength of SSL/TLS Servers: Current and Recent Practices. In *Proceedings of Internet Measurement Conference (IMC)*. 83–92.

[19] Xurong Li, Chunming Wu, Shouling Ji, Qinchen Gu, and Raheem Beyah. 2017. HSTS Measurement and an Enhanced Stripping Attack Against HTTPS. In *Proceedings of Security and Privacy in Communication Networks (SecureComm)*. 489–509.

[20] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. 1996. *Handbook of Applied Cryptography*. CRC Press, Inc.

[21] Shaun Nichols. 2018. Official: Google Chrome 69 Kills off the World Wide Web (in URLs). https://www.theregister.co.uk/2018/09/07/google_kills_www Accessed Dec. 1, 2018.

[22] OpenSSL Software Foundation. 2018. SSL_get_verify_result. https://www.openssl.org/docs/man1.1.1/man3/SSL_get_verify_result.html Accessed Dec. 24, 2018.

[23] MMXVIX. A Kenneth Reitz Project. 2019. Requests: HTTP for Humans. http://docs.python-requests.org/en/master Accessed April 3, 2019.

[24] Eric Rescorla. 2008. The Transport Layer Security (TLS) Protocol Version 1.2. https://www.ietf.org/rfc/rfc5246.txt Accessed Dec. 6, 2018.

[25] Eric Rescorla. 2018. The Transport Layer Security (TLS) Protocol Version 1.3 draft-ietf-tls-tls13-28. https://tools.ietf.org/html/draft-ietf-tls-tls13-28 Accessed Dec. 6, 2018.

[26] Liam Tung. 2018. Chrome 69 Kills Off www in URLs: Here's Why Google's Move Has Made People Angry. https://www.zdnet.com/article/chrome-69-kills-off-www-in-urls-heres-why-googles-move-has-made-people-angry Accessed Dec. 1, 2018.

[27] Serge Vaudenay. 2002. Security Flaws Induced by CBC Padding — Applications to SSL, IPSEC, WTLS..... In *Proceedings of Advances in Cryptology (EUROCRYPT)*. 534–545.

[28] W3Schools. 2019. Browser Statistics. https://www.w3schools.com/browsers Accessed Mar. 28, 2019.

[29] Matthias Wählisch, Robert Schmidt, Thomas C Schmidt, Olaf Maennel, Steve Uhlig, and Gareth Tyson. 2015. RiPKI: The Tragic Story of RPKI Deployment in the Web Ecosystem. In *Proceedings of Workshop on Hot Topics in Networks*.

[30] Yahoo Inc. 2016. tls-scan. https://github.com/prbinu/tls-scan Accessed Dec. 8, 2018.