Is the Web Ready for OCSP Must-Staple?

Taejoong Chung*
Rochester Institute of Technology

Jay Lok
Northeastern University

Balakrishnan Chandrasekaran
Max Planck Institute for Informatics

David Choffnes
Northeastern University

Dave Levin
University of Maryland

Bruce M. Maggs
Duke University and
Akamai Technologies

Alan Mislove
Northeastern University

John Rula
Akamai Technologies

Nick Sullivan
Cloudflare

Christo Wilson
Northeastern University

ABSTRACT

TLS, the de facto standard protocol for securing communications over the Internet, relies on a hierarchy of certificates that bind names to public keys. Naturally, ensuring that the communicating parties are using only valid certificates is a necessary first step in order to benefit from the security of TLS. To this end, most certificates and clients support OCSP, a protocol for querying a certificate's revocation status and confirming that it is still valid. Unfortunately, however, OCSP has been criticized for its slow performance, unreliability, soft-failures, and privacy issues. To address these issues, the OCSP Must-Staple certificate extension was introduced, which *requires* web servers to provide OCSP responses to clients during the TLS handshake, making revocation checks low-cost for clients. Whether all of the players in the web's PKI are ready to support OCSP Must-Staple, however, remains still an open question.

In this paper, we take a broad look at the web's PKI and determine if all components involved—namely, certificate authorities, web server administrators, and web browsers—are ready to support OCSP Must-Staple. We find that each component does not yet fully support OCSP Must-Staple: OCSP responders are still not fully reliable, and most major web browsers and web server implementations do not fully support OCSP Must-Staple. On the bright side, only a few players need to take action to make it possible for web server administrators to begin relying on certificates with OCSP Must-Staple. Thus, we believe a much wider deployment of OCSP Must-Staple is an realistic and achievable goal.

*This work was done while the author was a postdoctoral researcher at Northeastern University.

CCS CONCEPTS

• **Security and privacy** → **Public key (asymmetric) techniques**; • **Networks** → **Application layer protocols**; **Security protocols**; **Transport protocols**;

KEYWORDS

PKI; Public Key Infrastructure; HTTPS; OCSP

ACM Reference Format:
Taejoong Chung, Jay Lok, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce M. Maggs, Alan Mislove, John Rula, Nick Sullivan, and Christo Wilson. 2018. Is the Web Ready for OCSP Must-Staple?. In *2018 Internet Measurement Conference (IMC '18), October 31-November 2, 2018, Boston, MA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3278532.3278543

1 INTRODUCTION

Authentication—being able to verify with whom one is communicating online—is the foundation of secure communication. TLS, the authentication and encryption protocol underlying HTTPS, relies on *certificates* to bind servers' (and sometimes clients') identities to their public keys. Inevitably, keys are compromised and certificates are mis-issued, such as through vulnerabilities [10, 40, 41], poor implementations of cryptography [13, 38], or break-ins to certificate authorities' systems [2]. When this happens, it is critical that certificates be *revoked*.

Certificate revocation is therefore a critical component of any Public Key Infrastructure (PKI), and yet recent studies have found revocation to be woefully inadequate in the web's PKI. Website administrators revoke certificates at paltry rates [40, 41]; no browsers fully check for revocation information when connecting to TLS servers [22]; and certificate authorities host Certificate Revocation Lists (CRLs) that are untenably large [22]. When *any one* of these players fails to do their part, the revocation pipeline does not adequately protect users—unfortunately, today, all of them fail.

Historically, TLS clients have been required to check certificates' revocation status themselves during the TLS handshake using either a CRL or via the Online Certificate Status Protocol (OCSP). Not only does doing so cause a delay at the client—a client cannot continue the handshake until it knows that a certificate has not been revoked—it also potentially exposes the client's browsing behavior to the CRL or OCSP server. In an attempt to address some of the shortcomings of disseminating certificate revocation information, *OCSP Stapling* was proposed in RFC 6961 [27]. The idea behind OCSP Stapling is to have web servers periodically obtain OCSP responses and piggyback ("staple") these responses onto certificates as part of the TLS handshake to the client—thereby mitigating the privacy and latency costs inherent in CRLs and OCSP.

Unfortunately, OCSP Stapling does not completely solve the problem of certificate revocation, as clients can choose to continue a connection if an OCSP response is not provided (and prior work has shown that all popular web browsers do [22]). To address this challenge, certificates can include an *OCSP Must-Staple* extension [14], which informs the client that it *must* receive a valid OCSP response as part of the TLS handshake, or it should reject the certificate. OCSP Must-Staple is promising, and has the potential to greatly simplify and speed up the client side of revocation checking.

But, OCSP Must-Staple is not without challenges. For it to work, *all* players must do their part:

- **Certificate authorities** must provide OCSP responders that are globally and pervasively available, and that return valid and accurate OCSP responses;
- **Clients** (browsers) must advertise their support for stapled OCSP responses during the TLS handshake, and respect the OCSP Must-Staple extension by rejecting a certificate if it is not provided with a valid OCSP response;
- **Web server software** must be updated to support OCSP Stapling by properly caching OCSP responses and handling errors when communicating with OCSP responders;
- **Web server administrators** must support OCSP Stapling and must periodically request fresh OCSP responses.

In this paper, we study whether today's web is ready for OCSP Must-Staple. Specifically, we measure each of the three major principals—web servers, OCSP responders, and browsers—to ascertain whether they are doing what would be necessary for OCSP Must-Staple to succeed, and what impact their failures would have on website availability. Our study extends a preliminary study on the availability of OCSP servers [32] by measuring over more time, from more vantage points, and incorporating more of the principals who comprise the PKI.

Our main findings and contributions are as follows:

- While OCSP Must-Staple became an official RFC in October 2015, deployment remains low: only 0.02% of all certificates and 0.01% of certificates from Alexa Top-1M sites use it. However, the current most-popular CA, Let's Encrypt, now supports OCSP Must-Staple, meaning many domains can request such certificates.
- While examining the availability of OCSP responders, we find a number of outages during our four months measurement period. We also find a number of responders that return invalid responses, responses with superfluous certificates, and responses whose "close" validity time may cause clients with slightly slow clocks to consider the response invalid.
- We test all major web browsers and find *only one* (Firefox) currently supports OCSP Must-Staple. Unfortunately, the support is incomplete, as the iOS Firefox app does not yet support it.
- We test two popular web server implementations: Apache and Nginx. We find that neither fully or correctly supports OCSP Stapling, meaning clients may fail to be returned a valid OCSP response even though one may be available.

Overall, our results suggest that if just a few players—namely large CAs, popular browser vendors, and web server software maintainers—improved their support for OCSP Must-Staple, it could be widely deployed by server administrators and improve the overall security of the web's PKI.

The rest of this paper is organized as follows. Section 2 provides background on certificates, revocation mechanisms, and OCSP Must-Staple. Section 3 describes related work, and Section 4 provides a brief look at how widely deployed OCSP Must-Staple is today. Sections 5, 6, and 7 examine how well certificate authorities, clients, and servers, respectively, are doing what is necessary to support OCSP Must-Staple. Section 8 provides a concluding discussion.

Ethical considerations At all times, we took steps to ensure our measurements met community ethical standards. We responsibly disclosed our findings to all of the OCSP responders and their CAs that have service unavailabilities, potential security vulnerabilities (e.g., outages of OCSP servers, premature `thisUpdate` values, or unset `nextUpdate` values), discrepancies between the CRL and OCSP responses to help them mitigate the issues.[1] We also reported our findings to the software vendors.

2 BACKGROUND

In this section, we provide a brief background on certificates and detail the protocols for certificate revocation.

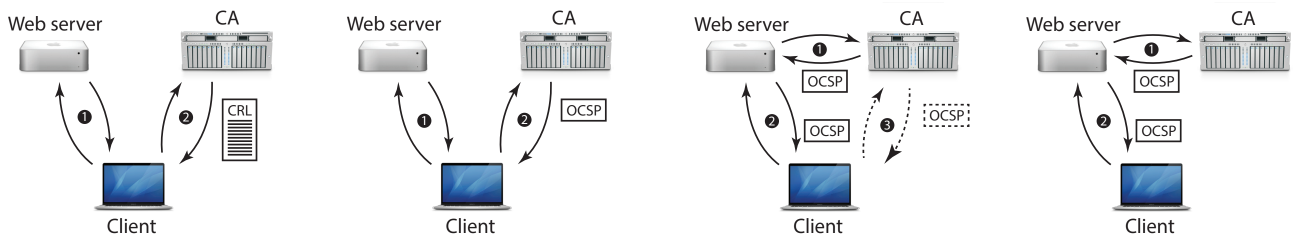
2.1 Certificates

A *certificate* is a signed attestation that binds a *subject* to a *public key*; in the web, the subjects are domain names. Certificates are typically issued by *certificate authorities* (CAs), who in turn have their own certificates signed by other certificates, terminating at a small set of self-signed root certificates.[2] Thus there is a logical chain of trust, starting from a root certificate through zero or more *intermediate* certificates, to a *leaf* certificate. To verify a certificate, a client therefore needs to obtain this chain of certificates and check that each has a correct signature, has not expired (each certificate has a well-defined validity period), and has not been revoked.

On the web, the format for a certificate is X.509 [5], which uses ASN.1 [8] to encode certificate data. X.509 certificates include at least one common name (the subjects), a public key, a serial number (unique for each issuer), a validity period, and directions for how to check if the certificate has been revoked.

[1] We provided them with CRLs we downloaded, OCSP request, and OCSP responses for their validation.

[2] Clients are assumed to obtain these root certificates out-of-band (e.g., most well-known browsers or operating systems maintain a set of root certificates).



(a) CRL (b) OCSP (c) OCSP Stapling (d) OCSP Must-Staple

Figure 1: Steps in the process of checking revocation status with different protocols: (a) with CRLs, the client fetches the (potentially large) CRL after obtaining the certificate in the TLS handshake; (b) with OCSP, the client asks for the revocation status of only the particular certificate; (c) with OCSP Stapling, the server is supposed to prefetch the OCSP response and provide it in the handshake, and if it does not, the client can fetch the OCSP response as in (b); and (d) with OCSP Must-Staple, the server *must* provide an OCSP response in the handshake or the client will reject the certificate.

2.2 Revocation: CRLs and OCSP

The owner of a certificate can ask its CA to revoke its certificate for a variety of reasons, such as usage of a cryptographically weak key [38], erroneous issuance, or key compromise [10, 40]. In such cases, the CA revokes the certificate by producing a public attestation that the certificate should not be trusted; this attestation is signed by the CA's private key. The CA is then responsible for disseminating such attestations for all revoked certificates that they have issued. There are two primary protocols by which clients communicate with CAs to obtain revocation attestations: Certificate Revocation Lists (CRLs) [5] and the Online Certificate Status Protocol (OCSP) [34]. We describe these below, and an overview is provided in Figure 1.

CRLs are ASN.1-encoded files that contain a list of (serial number, revocation timestamp, revocation reason) for all revoked certificates for a CA. CAs include a URL in a certificate as a `CRL Distribution Points` extension, so that a client can locate the CRL. Similar to X.509 certificates, each CRL contains a validity period (`Last Update` and `Next Update`) that specifies the range of time that it is good for; CAs are responsible for publishing CRLs regularly to update the validity period even if no additional certificates have been revoked.[3] Once a client has a fresh copy of a certificate's CRL, the client can quickly determine whether the certificate is revoked by checking if the certificate's serial number appears in the CRL. CRLs are often criticized, however, due to their inefficiency: a client needs to download *all* certificate revocation information even if it is only interested in the revocation status of a single certificate. In fact, prior work [22] has demonstrated that CRLs can be up to 76 MB!

OCSP is a web service protocol that allows a client to query the CA for the revocation status of a single certificate. The CA runs a server called an *OCSP responder* that answers such queries. Clients locate the OCSP responder for a certificate by using a URL that is provided in the certificate's `Authority Information Access` (AIA) extension. Each OCSP request must contain a given certificate's serial number along with a hash of the issuer's name and public key so that CAs can verify that they issued the certificate before responding. The OCSP responder returns a signed response that includes the following information:

- `certID`: the serial number of the queried certificate,
- `thisUpdate` and `nextUpdate`: the range of time for which the response is valid (i.e., how long it can be cached),
- `producedAt`: the time at which the OCSP responder generated this response, and
- `certStatus`: the certificate's revocation status, which is one of
  - `Good`: indicating that the certificate is not revoked;[4]
  - `Revoked`: indicating that the certificate has been revoked, either temporarily or permanently, or was never issued by this CA; or
  - `Unknown`: indicating that the responder does not know about the certificate being requested, typically because the certificate is not served by this responder.

OCSP responses are signed, and can be verified using the same public key that signed the requested certificate. However, OCSP responses can also contain a *leaf* certificate (itself signed by the same CA) that can be used to verify the signature of the OCSP response instead; this is called OCSP Signature Authority Delegation. Clients who receive a `Revoked` OCSP response are supposed to reject the certificate, while clients who receive an `Unknown` response are free to try another source of revocation information (e.g., another OCSP responder or a CRL) [34].

OCSP reduces the overhead of CRLs by allowing clients to query the CA for the revocation status for a single certificate. OCSP, however, has multiple issues; first, because clients depend on the OCSP response, the OCSP responders need to provide responses with low latency and high availability; second, one concern about OCSP is that CAs who run OCSP responders can observe much of the users' browsing behavior by monitoring OCSP requests, a potential privacy risk.

[3] To reduce the size of CRLs, CAs may remove revoked certificates from CRLs once they are expired.

[4] `Good` does not necessarily mean that the certificate is within its validity interval; a client must still check to make sure the certificate itself is still within its validity period.

2.3 OCSP Stapling

OCSP Stapling was introduced to address the additional latency that making OCSP requests engenders. With OCSP Stapling, the web server obtains an OCSP response ahead of time from the CA, and then provides this OCSP response to the client during the TLS handshake. As a result, a client receives *both* the server's certificate, its chain, and a (fresh) OCSP response at the same time, allowing the client to determine that certificate's revocation status with no additional network requests. Moreover, since the client does not need to connect to the CA's OCSP responder, OCSP Stapling alleviates the privacy concerns of OCSP.

Unfortunately, OCSP Stapling does not solve all issues with OCSP: *First*, a client needs to check the revocation status of *all* certificates on the chain using OCSP, but OCSP Stapling only allows the revocation status for the *leaf* certificate to be included. There is an extension to OCSP Stapling [27] that tries to address this limitation by allowing the server to include multiple certificate statuses in a single response, but it has yet to see wide adoption. *Second*, and more importantly, most clients (browsers) will accept a certificate even if they are unable to obtain revocation information via OCSP or CRLs [22]; this behavior is called "soft-failure" (as opposed to "hard-failure", where the client rejects the certificate if it cannot obtain revocation information [18]). If a client chooses to soft-fail in this case, an attacker who has control over the client's network could block any outgoing OCSP requests (and strip any stapled OCSP responses), thereby coaxing the client into accept a revoked certificate.

2.4 OCSP Must-Staple

OCSP Must-Staple aims to solve the problem of soft-failure: it is an X.509 certificate extension [14] that tells a client to *require* an OCSP response be provided (stapled) in the TLS handshake whenever it sees the certificate. If included in a certificate,[5] this extension acts as an explicit signal to the client that it must hard-fail if the server does not provide a fresh, valid OCSP response in the handshake.

In order for OCSP Must-Staple to be deployed successfully, the following entities need to take action:

(1) **Certificate authorities** must (a) include the OCSP Must-Staple extension into certificates they issue, with domain owners' consent, and (b) run highly available, correct OCSP responders to provide OCSP responses to web servers; and
(2) **Clients** (e.g., browsers and TLS libraries) must be updated to (a) understand the OCSP Must-Staple extension in certificates, (b) present the Certificate Status Request (CSR) extension to the web servers during the TLS handshake, and (c) reject the certificate if they do not receive OCSP responses from the web server; and
(3) **Web server software maintainers** must fully and correctly support OCSP Stapling, properly fetching and caching OCSP responses as well as handling errors when communicating with OCSP responders.
(4) **Web server administrators** must configure their servers to use OCSP Stapling.

[5]Each extension has its Object Identifier (OID), which of OCSP Must-Staple is 1.3.6.1.5.5.7.1.24

In the remainder of the paper, we look at each one of these entities in turn to see how close we are to being able to deploy OCSP Must-Staple today.

3 RELATED WORK

In this section, we discuss related studies aimed at understanding the web's certificate ecosystem and improving current revocation mechanisms.

Web certificate ecosystem There is a long thread of studies examining the web's certificate ecosystem, based on collections of certificates gathered from full IPv4 address scans or Certificate Transparency (CT) logs [11, 37]. Studies have specifically examined how certificates are managed in practice from the web administrator side [6, 7] and CA side [15], as well as in response to specific security vulnerabilities like Heartbleed [10, 40]. Other studies have measured the deployment of new security features such as CAA [33], SCSV, and others [3]. Many groups have proposed new techniques to improve the certificate ecosystem [19, 24, 29].

Our study complements these prior works. We study one of the most widely used revocation protocols in the certificate ecosystem, examining how it has been deployed and also discuss the challenges to improving it.

Challenge of managing revocation There have been a significant amount of related work on managing revocation information.

The size of revocations Certificates can be revoked due to a variety of reasons such as erroneous issuance and key compromises. The number of revoked certificates has grown (and continues to grow) significantly, making distributing revocation information to clients a challenging task. Prior work has attempted to address this issue by reducing the size of the revocation data or more efficiently disseminating this information. Topalovic et al., for instance, proposed short-lived certificates to potentially reduce the size of revocation data [35]; short-lived certificates might be more likely to expire than be revoked, and clients simply reject expired certificates. Recently, Larish et al. used a filter cascade to compress the revoked certificate lists [20]. Schulman et al., in contrast, proposed a new method for disseminating revocation data over FM radio [30].

The latency of revocation checks Maintaining an extensive list of revoked certificates imposes a huge burden on both CAs and clients—i.e., the onus of managing the list on CAs and of downloading such large lists, frequently, for revocation checks on clients. Although the OCSP protocol was introduced to alleviate this issue, it introduces additional delay in the TLS handshake for the clients. Several studies attempted to measure OCSP lookup latency and suggest improvements [17, 31, 39]. Stark et al. observed that the median latency for OCSP checks is 291 ms in 2012 [31]. In 2016, Zhu et al., however, reported a median latency of 20 ms—a significant improvement due to 94% of the requests being fronted by CDNs, according to the study [39]. The efforts of CAs and website administrators are, however, worthless if clients do not perform OCSP lookups or utilize the lookup response. Liu et al. focused on the revocation checking behavior of web browsers and operating systems and showed, for instance, that browsers often do not bother to check whether certificates are revoked; they showed that mobile browsers (iOS, Android, and IE) never checked for revocation [22]. Our study is complementary: We look at all different components (i.e., CAs, clients, and web servers) playing a critical role in the OCSP ecosystem to understand its current status and determine what they need to do for a widespread deployment of OCSP Must-Staple.

4 STATUS OF OCSP MUST-STAPLE

We begin by briefly examining the current deployment status of OCSP Must-Staple by identifying certificates issued with the OCSP Must-Staple extension.[6] For this section, we obtain our collection of certificates from Censys [9], which aggregates certificates using both *full* IPv4 port 443 scans and public Certificate Transparency servers [19]. We use the Censys snapshot of this dataset that was collected on April 24th, 2018, which contains 489,580,002 certificates. Consistent with prior work [7], we focus only on a set of 112,841,653 valid certificates that were marked as trusted by Censys.[7]

We first examine the fraction of valid certificates that support OCSP—a necessary pre-condition for supporting OCSP Must-Staple. We do so by checking for the existence of at least one OCSP URL in the certificates' AIA extension. We find that 107,664,132 certificates (95.4% of the 112,841,653 valid certificates) support OCSP, confirming the prevalence of OCSP noted in prior work [22]. When we examine further to see how many certificates also support OCSP Must-Staple, however, we discover only 29,709 (0.02%) certificates, indicating that the deployment of OCSP Must-Staple is still quite low. From each of these 29,709 certificates, we obtain the issuing CA's URL from the `Certificate Authority Information Access` extension to determine which CAs support OCSP Must-Staple today. Surprisingly, we find that 28,919 (97.3%) of the certificates that support OCSP Must-Staple are issued by Let's Encrypt[8] while the remaining are issued by Comodo (73), DFN (716), and UserTrust (1).

Next, we quickly check whether "popular" domains are more likely to have deployed OCSP Must-Staple. Figure 2 shows (1) the percentage of the Alexa Top-1M domains [4] that support HTTPS, and (2) the percentage of those domains that support OCSP. We observe that HTTPS support for popular websites is now close to 75% across the entire range, that OCSP adoption is quite high (91.3% on average), and that popular domains are (slightly) more likely to support OCSP. Unfortunately, we find that only 100 certificates (0.01%) from the Alexa Top-1M domains support OCSP Must-Staple—OCSP Must-Staple has a long way to go before it is widely deployed.

In the next section, we explore why OCSP Must-Staple has not been widely deployed and what needs to be done to encourage its deployment by CAs, clients, and web server administrators.

[6]Note that we are not yet examining whether the web servers actually *provide* an OCSP response in the TLS handshake; we will see this later in Section 7.

[7]To validate the certificates, Censys uses the Apple, Microsoft, and Mozilla NSS root stores; we consider the certificate if it is valid using at least one of those three root stores.

[8]Let's Encrypt began to support OCSP Must-Staple in May, 2016 [23].

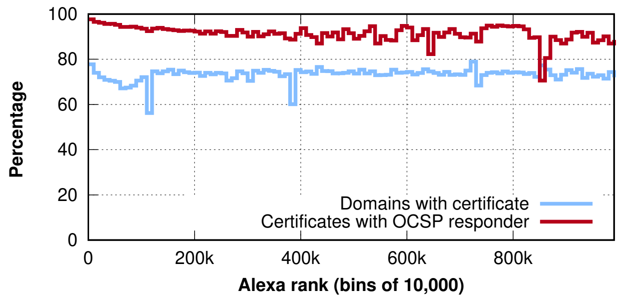


Figure 2: OCSP adoption as a function of website popularity; the most popular websites that support HTTPS tend to do OCSP as well.

5 CERTIFICATE AUTHORITIES

CAs are responsible for running highly available, accurate OCSP responders that can provide OCSP responses to web servers for use in OCSP Stapling. If these OCSP responders are not reliable, web servers will be unable to staple OCSP responses during the TLS handshake—in an OCSP Must-Staple world, clients (browsers) would refuse to connect to these web servers. Running highly-reliable OCSP responders is therefore critical to the successful deployment of OCSP Must-Staple. In the remainder of this section, we evaluate the availability and reliability of OCSP responders before turning our attention to the "quality" of OCSP responses.

5.1 Methodology

Our goal is to understand how reliable CAs' OCSP responders are today. To this end, we collected two data sets, one focusing on a random sampling of *all* OCSP responders and the other focusing only on the OCSP responders for certificates of popular domains.

Complete OCSP responders scans The following steps detail our methodology for investigating how *all* OCSP responders behave.

(1) We first extracted all *valid* certificates from the Censys dataset (described in Section 4) that had at least 30 days of validity remaining. This step yielded 77,399,894 certificates.
(2) We grouped the certificates by OCSP responder, which was retrieved from the certificates' AIA extension. Each grouping represents all of the certificates served by a single responder. For certificates with multiple OCSP responders (6,308 in total or 0.008%), we choose one responder at random.
(3) We selected 50 random certificates for each OCSP responder, and for OCSP responders associated with less than 50 certificates, we picked all the certificates.
(4) We developed a measurement client that issued OCSP requests using the HTTP POST method for all of the selected certificates (i.e., we performed an OCSP lookup for each certificate against its corresponding OCSP responder) every hour.[9]
(5) We deployed our measurement client in six different vantage points around the world—Oregon (Amazon Web Services

[9]During our measurement period, we excluded certificates from our measurement results once they had expired.

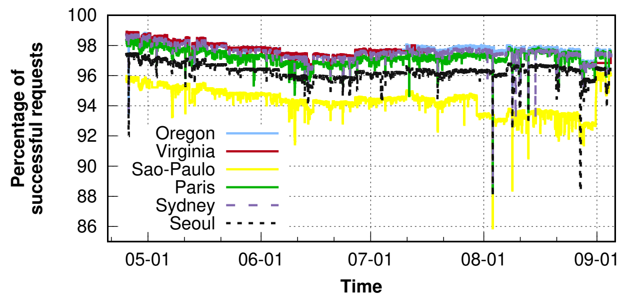


Figure 3: Fraction of requests that result in a successful response for the Hourly dataset, for each of our measurement clients.

[AWS] U.S. West), Virginia (AWS U.S East), São Paulo (AWS Brazil), Paris (AWS France), Sydney (AWS Australia), and Seoul (AWS South Korea)—to obtain a comprehensive understanding of how responders behave.

We used the above methodology to gather measurements from 536 OCSP responders by issuing 14,634 OCSP requests (i.e., lookup the revocation status of 14,634 certificates) every hour from April 25, 2018 to September 4, 2018; we refer to these measurements as the Hourly dataset.

Alexa Top-1M Scan While the Hourly dataset suffices for studying OCSP responders' behavior in general, it does not capture the behavior of OCSP responders of popular domains' certificates. To address this limitation, we collected a second dataset by sending OCSP requests to all domains in the Alexa Top-1M list [4] that support HTTPS and OCSP.

Of the Alexa Top-1M domains, we identified 606,367 certificates corresponding to the domains that support HTTPS and OCSP in their root certificates. Using the same six vantage points used for the Hourly dataset, we perform OCSP lookups once for these 606,367 certificates, measuring responses from 128 OCSP responders, on May 1st, 2018. We refer to this data set as the Alexa1M dataset.

In the remainder of this section, we utilize these two datasets to examine the availability and reliability of today's OCSP responders.

5.2 Availability

To support OCSP Must-Staple, CAs need to run OCSP responders that are highly available to provide OCSP responses to web servers. We begin by examining how reliable their OCSP responders are over time. We first focus on the portion of OCSP responses where we are unable to successfully interact with the OCSP responder. As OCSP requests are sent over HTTP, we define a *successful request* as a request that resulted in the server responding with HTTP status code `200`. Reasons for unsuccessful requests can include DNS failures, inability to connect to the OCSP responder, HTTP status codes other than `200`, etc. Note that a successful request does not by itself imply that a *correct OCSP response* was obtained; there could be malformed responses, invalid signatures, etc. We examine all of these errors in subsequent sections. Figure 3 plots the fraction of requests that were successful from the Hourly dataset from each of our six different vantage points; we make a number of observations.

First, we observe that we were *never* able to receive successful requests from *all* OCSP responders in a given hour in any of our measurement client locations. On average, 1.7% of requests failed; in fact, for two OCSP responders,[10] we were never able to make a successful OCSP request from any of our six vantage points. This implies that clients who are served certificates making use of these responders would *always* fail to be able check the revocations status of all certificates in the chain. For 29 other responders, there was at least one measurement client that was never able to make a successful request. Looking into our logs, we find a variety of reasons why there were persistent failures:

- For 16 responders, we observed persistent DNS lookup failures (`NXDOMAIN`) from at least one client;
- For 4 additional responders, we were never able to establish a TCP connection to them from at least one client;
- For 8 more responders, we persistently received HTTP `4xx` or `5xx` response codes from at least one client;
- Finally, for 1 responder, at least one client was unable to connect to the HTTPS URL because it was served with an invalid certificate.

Second, the failure rate varies substantially across different locations: the average failure rate ranges between 2.2% (Virginia) and 5.7% (São Paulo) of requests. We find that the measurement clients located at Oregon, São Paulo, Paris, and Seoul always fail to fetch OCSP responses from one, seven, one, and four responders, respectively. For example, five OCSP URLs are subdomains of *.digitalcertvalidation.com, all of which return HTTP 404 errors to our measurement client located in São Paulo; statush.digitalcertvalidation.com is one of those URLs. Unfortunately, a certificate of wellsfargo.com, which is one of the largest banks in the United States, relies on this OCSP URL; hence, any client in São Paulo would not be able to fetch the certificate revocation status of wellsfargo.com from OCSP servers even if the certificate were compromised and revoked.[11]

Third, we observe multiple transient outages, which usually last a couple of hours. During our measurement period, we observed that 211 (36.8%) OCSP responders experienced at least one outage from at least one vantage point.[12] For example, we notice that all of our OCSP requests made to `ocsp.comodoca.com` failed at 7pm, April 25 for two hours. Interestingly, this outage was observed only at the clients in Oregon, Sydney, and Seoul. We also found that an additional 14 OCSP responders experienced outages at the same time, all of which were related to Comodo: the domain names of eight OCSP responders had `CNAME` records that pointed to `ocsp.comodoca.com`, and the domain names of the remaining six OCSP responders resolved to the same IP address as `ocsp.comodoca.com`. Similarly, we found that all of our OCSP requests to the servers managed by `wosign` and `startssl` failed at 10pm, August 3 for an hour across

[10]https://ocsp.IdenTrustSAFEca1.identrust.com, https://ocsp.IdenTrustSAFERootca2.identrust.com

[11]These OCSP servers were fixed at 11pm, August 31.

[12]Interestingly, we observed decreasing trends of the percentage of successful requests for the first month (Figure 3). This was due to that some OCSP servers such as http://ocsp.pki.wayport.net:2560 had become unavailable gradually during that time.

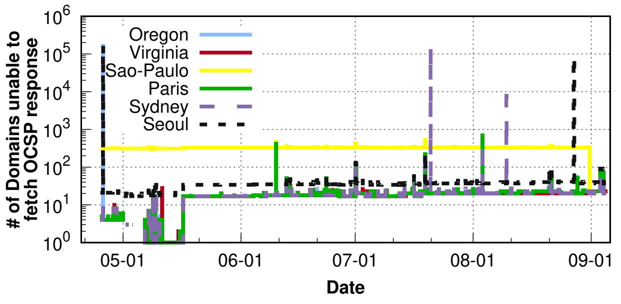


Figure 4: The number of domains in Alexa Top-1M where we were unable to obtain a successful response. Due to the outage of `ocsp.comodoca.com`, 25% of domains were unable to provide OCSP responses during at least two hours.

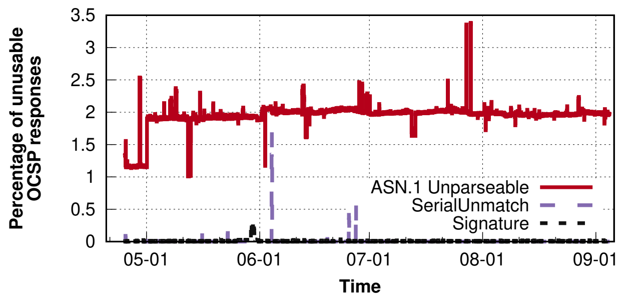


Figure 5: The percentage of OCSP responses, which are not able to use due to (1) malformed OCSP structure (ASN.1 structure error), (2) serial unmatch, and (3) signature validation fails

the regions. Interestingly, some outages were only observed at the specific regions; 9 servers managed by Digicert[13] were down at 9am, August 27 for 5 hours, which was only observed at the client in Seoul. Similarly, all of our OCSP requests made from the clients in Sydney to 16 OCSP servers managed by Certum failed at 5pm, August 9 for two hours.

Impact of Outages Popular domains' certificates are usually issued by a small number of CAs [37], meaning their OCSP responders are likely hosted by a small number of entities. This centralization potentially creates a small number of points of failure: an outage of a popular OCSP responder could make many certificates unavailable for their revocation check. We now use our Alexa1M dataset to infer how many popular domains had certificates with unavailable OCSP responders, allowing us to estimate the *impact* of OCSP responder unavailability.

Figure 4 shows the number of top domains that support OCSP where clients trying to check the revocation status of that domain's certificate were unable to make a successful request during the measurement period. Surprisingly, almost 163K domains were not able to provide OCSP responses to the clients in Oregon, Sydney, and Seoul due to the outage of their OCSP responders on April 25, 2018 for two hours. This was mainly due to the outage of OCSP servers managed by `Comodo` as mentioned in the previous analysis. Similarly, 77K domains were not able to provide OCSP responses to the clients in Seoul due to the outage of 9 Digicert OCSP servers on August 27 for 5 hours. We also observe that the client in São Paulo is always unable to fetch the OCSP responses of 318 (0.05%) domains' certificates.

CDN's Perspective CDNs, which are used by certificate authorities to cache OCSP responses to improve scalability and reliability, frequently contact OCSP responders. To obtain a CDN's perspective on the availability of OCSP responders, we collected logs from Akamai's CDN servers deployed at two locations that serve a substantial volume of TLS traffic. The logs, spanning a period of approximately 60 hours, reveal that the CDN contacts a small number of OCSP responders (approximately 20) compared to our active measurements (Alexa1M dataset). Because most responses are served from cache, only a small fraction of TLS connections, unsurprisingly, cause the CDN servers to contact the OCSP. But in those instances in which the CDN servers contacted OCSP responders, the HTTP status codes recorded in the logs indicate a 100% success rate.

[13]status(a|d|e|g|h).digitalcertvalidation.com, ocsp.digicert.com, ocsp(1|2|x).digicert.com

5.3 Validity

Even though our measurement client is able to make a successful request, the response might be invalid due to a number of reasons, including

- **Malformed structure**: the client is unable to parse the OCSP response if it does not follow the ASN.1 specification [34].
- **Serial number mismatch**: the serial number of the certificate in the OCSP response does not match the serial number that our client requested.
- **Incorrect signature**: the signature in the OCSP response is unable to be verified using (1) certificates in the OCSP response or (2) the issuer's certificate.

Figure 5 shows the distribution of the errors during our measurement period. We notice that the vast majority of the errors are caused by a malformed structure of the response, i.e. no responses that are correctly formed have invalid signatures or mismatched serial numbers. Examining the malformed responses, we find eight (1.6%) OCSP responders that persistently malformed responses, including empty responses, the value "0", or even JavaScript pages. Additionally, we observed a spike in the malformed responses on April 29, 2018 that lasted for 6 hours; this was due to 6 OCSP responders from `*.sheca.com` misbehaving and returning the response "0" for all requests. They returned those unusable responses again at 5pm on July 28 2018 for 3 hours. We also found 3 OCSP responders from `postsigum.cz` that began returning "0" responses for all requests on May 1st, 2018. The issue disappeared at 9am on May 12th for 17 hours, but began returning "0" responses again after then.

5.4 Quality

Finally, we turn to examine the "quality" of the OCSP responses. Thus far, we have found that we are able to make successful requests to most OCSP responders, and that most of those responders return

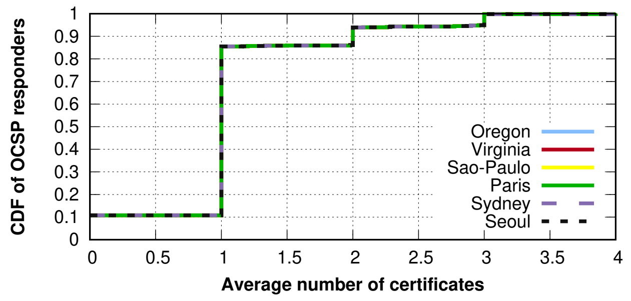


Figure 6: Cumulative distribution of average number of certificates in an OCSP response by OCSP responder; 79 (15%) of responders *always* put more than one certificate in their OCSP responses.

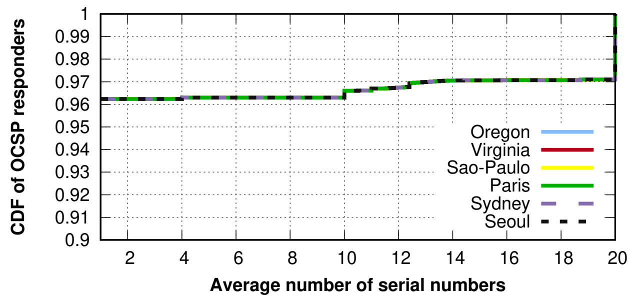


Figure 7: Cumulative distribution of average number of serial numbers in an OCSP response. Note that x axis starts from 1 and y axis starts from 90%.

valid responses. However, there are a still a number of ways in which OCSP responses could not meet best practices; we outline those below.

Superfluous certificates and responses OCSP responses usually contain a single leaf certificate signed by the same CA to allow clients to verify the OCSP response. However, the OCSP specification allows the responder to include additional certificates to help validation (even though those certificates should already be on the client, as they are necessary to validate the certificate that is being asked about). Thus, if OCSP responders include such certificates, they typically only serve to make the size of the OCSP response bigger. Figure 6 shows the cumulative distribution of the number of the certificates in an OCSP response from the Hourly dataset; we notice that 14.5% of the responders are sending OCSP responses that contain more than one certificate. For example, an OCSP responder, `ocsp.cpc.gov.ae`, always put four certificate chains including the root certificate in the OCSP responses, increases OCSP response size and the time it takes for clients to parse the response.

Similarly, OCSP requests typically concern only a single certificate (serial number). However, the OCSP specification allows OCSP responders to include OCSP responses for other, unsolicited certificates as well; doing so serves to inflate the response. Figure 7 shows the cumulative distribution of the number of the serial numbers in one OCSP response from the Hourly dataset. Most of the

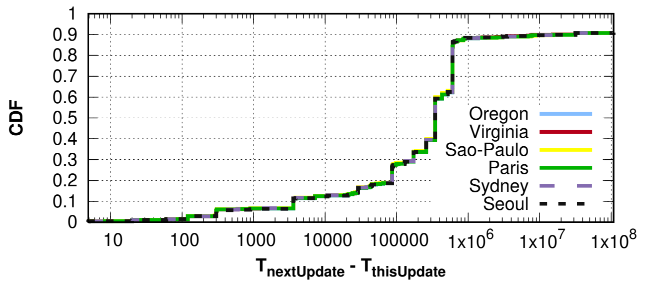


Figure 8: Cumulative distribution of validity period; note that x extends to 108,130,800 seconds, or 1,251 days!

OCSP responders (96.2%) put just one serial number in one OCSP response, but 4.8% of them put more than one serial number in the response. 17 (3.3%) responders always put *20* serial numbers in each response.

Validity period The `thisUpdate` and `nextUpdate` times are a critical component of OCSP responses, as they define when OCSP responses are valid; clients will reject OCSP responses if they have expired—or if they have yet to be valid—even if they were obtained from the OCSP responder with valid format, signatures, and correct serial numbers.

Validity Period We refer the period between the `thisUpdate` and `nextUpdate` times as the response's *validity period*; this is similar to the validity period of X.509 certificates. However, unlike the validity period of X.509 certificates, OCSP responders can set the `nextUpdate` to be *blank* to indicate that newer revocation information is always available; this encourages clients not to cache the OCSP response. As their expiration date does not exist, however, it is technically *always* regarded as valid, which could potentially raise security vulnerabilities with cached responses.

Figure 8 shows the cumulative distribution of average validity periods for each OCSP responder; note that we regarded the validity period as infinite seconds when we observe the blank `nextUpdate`.[14] We immediately notice that the validity periods are consistent over six different vantage points, suggesting a consistent policy by CAs that have a distributed deployment of OCSP responders. We also observe that 45 OCSP responders (9.1%) always set their `nextUpdate` to be blank, indicating that they can always provide newer revocation information, which could potentially increase their incoming workloads as clients may choose to *not* cache their responses. Surprisingly, we also notice that 11 OCSP responders (2%) set their validity periods over one month, which is potentially dangerous: if the certificate were compromised, there could be some clients who cache the previous response and would not obtain a fresh revocation status for up to 1,251 days!

Premature `thisUpdate` *values* The `thisUpdate` and `nextUpdate` define the validity period of OCSP responses[15]; hence, it is critical

[14]We observe consistent behavior by the OCSP responders that set `nextUpdate` to be blank; they *always* do so for all their responses.

[15]Similar to X.509 certificates, all time values in OCSP responses must be represented as Greenwich Mean Time (Zulu).

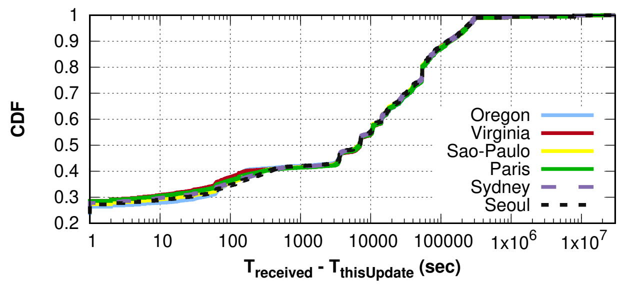


Figure 9: Cumulative distribution of the time between the `thisUpdate` and the time that we received the OCSP response across six different vantage points; we identified 85 (17.2%) OCSP responders have returned the response without giving any margin to their `thisUpdate` regardless of the vantage points.

for clients and servers to have synchronized clocks to correctly determine whether the certificate is valid or not. If client clocks are behind the `thisUpdate` value set by the OCSP responder, then clients will reject the OCSP response as it is *yet to be valid.* Figure 9 shows the time difference between the `thisUpdate` and the time that we received the OCSP response across six different vantage points; note that we synchronized the clock of clients correctly using NTP. Interestingly, we observe that 85 (17.2%) OCSP responders have returned the OCSP responses without having any margin of error in their `thisUpdate` (i.e., the response became valid at the same time our measurement client made the request); 15 (3%) of OCSP responders even returned OCSP responses with future `thisUpdate` times, which indicates that their response would be rejected as not yet valid.

Expired `nextUpdate` *values* We also looked for OCSP responses where the `nextUpdate` was set to be behind to the actual time clock (i.e., responses that had already expired); however, we did not find any instances of this behavior.

Non-overlapping validity periods OCSP responders may generate an OCSP response on demand when an OCSP request arrives, or may cache it and serve the same response over time until it expires for efficiency. In the latter case, they should update their OCSP responses before they expire; if they set their validity periods to be less than or equal to the period where they update the OCSP response, some clients would not be able to fetch fresh OCSP responses [32]. We examine how many OCSP responders could have this issue; we first filter the OCSP responders that generate the OCSP responses on demand. However, when receiving an OCSP response, we do not know whether the response has been generated on demand or not. To address this limitation, we use the `producedAt` value in OCSP responses; we only consider OCSP responses where the difference between `producedAt` and the time that we received the response is larger than 2 minutes, which indicates that the response has not been generated on demand. After filtering we examine (1) how frequently fresh OCSP responses are generated by calculating the difference between two `producedAt` values in two consecutive scans and (2) if their `validityPeriod` is less than or equal to those periods.

Among 483 OCSP responders that we are able to correctly fetch their time values, we find that 245 (51.7%) OCSP responders *do not* generate OCSP responses on demand; for those OCSP responders, we observe that 7 responders whose validity period is equal to the period that they update OCSP responses.[16] For example, 3 OCSP responders are subdomains of hinet.net, all of which set `validityPeriod` of their OCSP responses to 7,200 seconds and update them every 7,200 seconds. Similarly, a responder from ocspcnnicroot.cnnic.cn sets the `validityPeriod` to 10,800 seconds and updates them at the same rate.[17]

Consistency between OCSP and CRL Both CRLs and OCSP allow clients to check the revocation status of certificates. If certificates support both extensions, clients are free to use either one of them (or both); however, regardless of whether or not they have been revoked, it is assumed that the results are consistent. We now briefly examine if this is actually the case.

Methodology Our goal in this section is to find if there are discrepancies between the revocation status of certificates in CRL and OCSP; to this end, we obtain a set of certificates that support both CRL and OCSP extensions. We first obtain unique CRLs by examining the Alexa Top-1M domains' certificates where they support both OCSP and CRLs; among the unique 1,579 CRLs used by this set, we find that certificates that use 1,568 of the CRLs also support OCSP. Thus, most CAs support both CRLs and OCSP.[18]

We then download all 1,568 CRLs, parse them, and obtain the set of revoked serial numbers and their timestamps; this set contains 2,041,345 different serial numbers. When trying to issuing OCSP requests with the serial numbers, we face a significant challenge: CRLs only contain serial numbers, revoked time, and (optional) revoked reasons, not the validity period of certificates. OCSP responders are allowed to return an `Unknown` response if the queried certificates has expired, potentially skewing our results. To address this limitation, we only consider the certificates that we know to be valid: we cross-reference these 2,041,345 serial numbers with the certificate dataset we introduced in Section 4, looking for matching serial numbers and issuers. We then disregard any certificates that appear in the CRLs but have already expired. From this process, we obtain 728,261 unexpired-and-revoked certificates that cover 1,193 CRLs; we issue OCSP requests for each of these certificates on May 1st, 2018 and we are able to collect 727,440 (99.9%) OCSP responses.

Discrepancy: Revocation Status If the revocation status of certificates is not consistent across CRL and OCSP, it would raise significant concerns; the revoked certificates might be treated as *valid* depending on which revocation protocol clients choose. We examine the revocation status from OCSP responses and CRLs to see if there are

[16]Fortunately, we do not observe any responders who update their responses less frequently than their `validityPeriod`.

[17]Interestingly, we often observe that the difference between `producedAt` values between consecutive scans of this responder goes negative every 3 or 4 scans; from manual investigation, we noticed that they run multiple OCSP responders sharing the same IP address. This indicates that clients might get stale OCSP responses depending on which responder they would connect to.

[18]Let's Encrypt is a notable exception, as it *only* supports OCSP.

OCSP URL	CRL	# of Certificates where the OCSP response is		
		Unknown	Good	Revoked
ocsp.camerfirma.com	crl1.camerfirma.com/camerfirma_cserverii-2015.crl	0	7	369
ocsp.quovadisglobal.com	crl.quovadisglobal.com/qvsslg3.crl	0	1	514
ocsp.startssl.com	crl.startssl.com/sca-server1.crl	0	1	980
ss.symcd.com	ss.symcb.com/ss.crl	0	1	28,023
twcasslocsp.twca.com.tw/	sslserver.twca.com.tw/sslserver/Securessl_revoke_sha2_2014.crl	0	1	122
ocsp2.globalsign.com/gsalphasha2g2	crl2.alphassl.com/gs/gsalphasha2g2.crl	5,375	0	0
ocsp.firmaprofesional.com	crl.firmaprofesional.com/infraestructura.crl	11	0	0

Table 1: Among 1,193 CRLs, we found 7 CRLs where at least one certificate in the CRL is *not* regarded as revoked when we send an OCSP request to the corresponding OCSP responder.

any discrepancies; surprisingly, we found seven OCSP responders that responded with *different* revocation status. Table 1 shows the CRLs and corresponding OCSP URLs that have discrepancies. We observe that five of these OCSP responders returned Good—and two responders returned Unknown—for at least one certificate that appears in that CA's CRL (i.e., is revoked according to the CRL)[19] In fact, one of these responders returned Unknown for *all* 5,375 certificates that appeared on the CA's CRL.

Discrepancy: Revocation Time It is important for CAs to update the revocation status of certificates in a timely manner upon revoking certificates; if it were updated later in either CRLs or OCSP, the clients who rely on the "late" revocation protocol would not be able to notice revocations until it is updated. We examine the difference between the revocation times in OCSP responses and CRLs.[20]

Figure 10 shows the cumulative distribution of the revocation time of certificates in CRLs subtracted from that of OCSP responses. First, we observe that only 863 OCSP responses (0.15%) have different revocation time, which indicates that the revoked time is usually updated simultaneously to CRLs and OCSP responders. Of those who have different revoked times, we notice that 127 (14.7%) of the revoked responses have negative time differences, which imply that the OCSP response is updated later. Interestingly, we find that all the revoked certificates' revoked time retrieved from one OCSP responder, `ocsp.msocsp.com` are behind the corresponding CRL by between 7 hours and 9 days!

Discrepancy: Revocation Reason Finally, when a certificate is revoked, both CRL and OCSP can contain a *reason code*[21] that is supposed to explain why the certificate was revoked. While the reason codes are not typically used by clients, we nevertheless examine the discrepancies of the revocation reason codes as an additional measure of consistency between CRLs and OCSP. We find more than 87,000 (15%) cases where revocation reasons differ; however, the vast majority (99.99%) is due to cases where the CRL contains a reason code but the OCSP server does not. This result aligns with the previous findings that the vast majority of the revocations actually include no reason code [21].

[19]We contacted all five of those CAs to report our findings; Quovadis and Camerfirma responded that they maintain two different databases for revocation status of CRL and OCSP server, which might cause inconsistent revocation status of certificates. More specifically, Quovadis said that those expired certificates were rejected upon insertion into the OCSP database due to max character size (e.g., the certificates with over 100 SAN Fields).

[20]As it is impossible to measure externally when the certificate is actually revoked, we assume that the revocation times specified both in CRLs and OCSP reflect when it is updated, which CAs are supposed to do; the revocation time is defined as the time at which the certificate was revoked [5, 34].

[21]Both CRLs and OCSP share the same set of revocation reason codes [5].

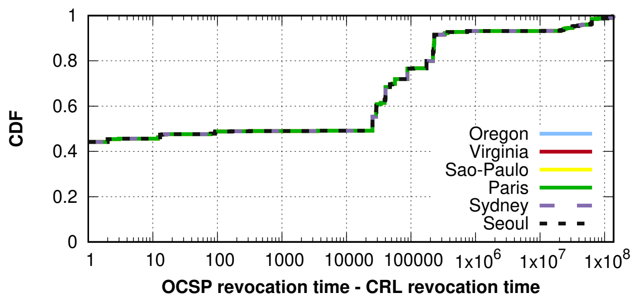


Figure 10: Note that x axis starts from -43,200 seconds and 14.7% of time differences are negative, which means that the revocation time in the OCSP response is earlier than that in the CRL. The long tail for the difference of revoked times extends to over 137M seconds (which is over 4 years!).

6 CLIENTS

We now turn to examine the role that clients (e.g., web browsers) play in deploying OCSP Must-Staple. To correctly support OCSP Must-Staple, clients must (1) solicit OCSP responses from the web server during the TLS handshake by adding the `Certificate Status Request` extension to the request, and (2) reject certificates that have the OCSP Must-Staple extension but were served without a valid OCSP response. In this section, we explore the extent to which the most popular TLS clients—web browsers—correctly support OCSP Must-Staple.

Methodology First, we purchase a domain name and obtain a valid certificate with the Must-Staple extension issued by Let's Encrypt. Then, we run the Apache web server and configure it to serve this certificate. However, we deliberately disable OCSP Stapling,[22] which prevents Apache from serving an OCSP response during the TLS handshake.

[22]We use the setting `SSLUseStapling off`.

	Desktop Browsers											Mobile Browsers				
	Chrome 66			**Firefox** 60			**Opera**		**Safari**	**IE**	**Edge**	**Safari**	**Chrome**		**Firefox**	
	OS X	Lin.	Win.	OS X	Lin.	Win.	OS X	Win.	11	11	42	iOS	iOS	And.	iOS	And.
Request OCSP response	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Respect OCSP Must-Staple	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Send own OCSP request	✗	✗	✗	-	-	-	✗	✗	✗	✗	✗	✗	✗	✗	✗	-

Table 2: Browser test results; all of the desktop and mobile browsers request OCSP responses in the SSL/TLS handshake (indicated as ✓); however, most of the browsers, except the Firefox on Android and across three desktop OSes, show warnings or terminate the connection even though they do not receive an OCSP response with an OCSP Must-Staple certificate (indicated as ✗). Also none of those browsers who accept the certificate make own OCSP request to the OCSP responder.

We then choose a variety of popular web browsers; Chrome, Firefox, Opera, Safari, Internet Explorer, and Microsoft Edge on desktop OSes (OS X, Linux, Windows) and mobile OSes (iOS and Android).[23] From each client, we try to connect to our test domain. While doing so, we capture all traffic generated from the client to ascertain whether it solicits an OCSP response by sending the `Certificate Status Request` extension in the TLS handshake [1]. If so, we then determine whether the client refuses to accept the certificate, makes its own OCSP request to the OCSP Responder[24], or simply accepts the certificate with no revocation information at all.

Results The results of this experiment are summarized in Table 2. We first observe that all browsers do solicit stapled OCSP responses, indicating that they support OCSP Stapling. Compared to the previous work from 2015 [22], we are able to confirm that Safari's behavior has changed, as it previously did not even request a stapled OCSP response. However, we observe that only Firefox (on all desktops OSes and on Android) displays a certificate error to the user if a stapled OCSP response is not included with a certificate that includes the OCSP Must-Staple extension; all other browsers (including Firefox on iOS) simply accept the certificate and do not even send their own OCSP request to the OCSP responder. These results indicate that clients are largely not yet ready for OCSP Must-Staple. However, it does appear that all clients already support OCSP Stapling, meaning the additional coding work necessary to support OCSP Must-Staple is likely not too significant.

7 WEB SERVER ADMINISTRATORS

As a final point of analysis, we turn to examine the role that web server administrators play in deploying OCSP Must-Staple. When certificates support OCSP Must-Staple, the OCSP response *must* come from the web server as part of the TLS handshake. Thus, these web servers are required to periodically fetch fresh OCSP responses from the OCSP responders to use when communicating with clients. In this section, we examine the current status of OCSP Staple support by web servers in-the-wild, and we test two well-known web servers (Apache and Nginx) to see how well they correctly support OCSP Must-Staple.

[23] All tests were done on Ubuntu 16.04, Windows 10, OS X 10.12.6, iOS 11.3, and Android Oreo.

[24] Note that Let's Encrypt certificates do not support CRLs, so no client will request CRLs in our experiments.

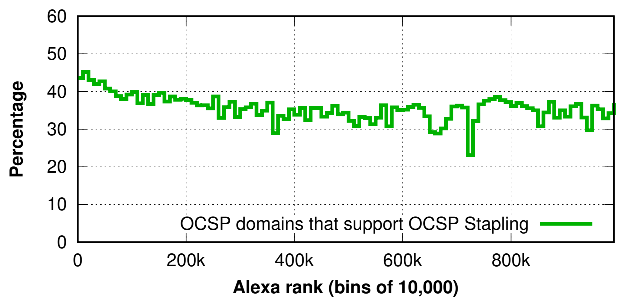


Figure 11: OCSP Staple adoption as a function of website popularity; the most popular websites that support OCSP tend to do OCSP Stapling as well.

7.1 OCSP Stapling support

We begin by examining how commonly OCSP Stapling is deployed by web server administrators today. A certificate by itself does not tell whether an administrator has enabled OCSP Stapling; instead, we need to see if the web server provides an OCSP response during the TLS handshake. To do so, we use the TLS handshake scans of Alexa Top-1M domains collected by Censys [9] that were collected on April 30, 2018. This dataset contains the complete logs of TLS handshakes with the root domain of all Alexa Top-1M domains. Figure 11 shows the percentage of domains with OCSP-enabled certificates that support OCSP Stapling. We observe that roughly 35% of the domains do so, and that popular domains are more likely to support OCSP Stapling.

Next, we briefly look at how this support has changed over time. To do so, we obtain the same TLS handshake scans of Alexa Top-1M domains from Censys [9] on a monthly basis going back to May 2016. Figure 12 plots the fraction of domains that support OCSP and those that support OCSP Stapling as well. We first notice that both fractions of (1) HTTPS domains that support OCSP and (2) those domains that also support OCSP stapling are steadily growing. Interestingly, we notice a spike in June 2017 that was due to Cloudflare. The number of domains that support OCSP Stapling and serve certificates containing one of the Cloudflare's domains[25] is 11,675 on May 18, 2017 but increases to 78,907 by June 15, 2017.

[25] These certificates are called "cruise-liner" certificates [6]; they contain many domains per certificate, which allows hosting providers to serve many domains.

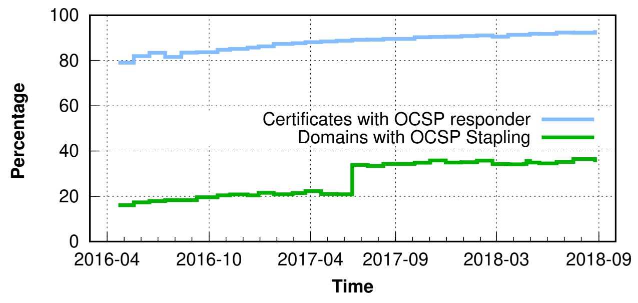


Figure 12: The percentage of OCSP and OCSP Stapling adoption from monthly-fetched Censys Alexa 1M datasets since May 21st, 2016. Note that the certificates that support OCSP Must-stale are not shown here as there are the only 100 certificates found in the latest snapshot.

7.2 Software Support

If we are to move towards a larger deployment OCSP Must-Staple, web server software *must* have a correct implementation of OCSP Stapling. In other words, web servers must (1) provide OCSP responses in TLS handshakes without causing extra delays to the clients, (2) cache OCSP responses and use them correctly considering their validity periods, and (3) cope with intermittent unavailability and errors of OCSP responders. However, it remains unclear whether web servers achieve these objectives [36].

In this section, we examine whether popular web server software *correctly* supports OCSP Stapling; more specifically, we focus on their behaviors in three perspectives:

- **Performance**: First, we see if web server software proactively fetches OCSP responses. If web servers do not—and instead fetch OCSP responses on-demand—this could either introduce unnecessary latency in completing the TLS handshake, or cause the web server to not include an OCSP response when responding to the first (few) client(s).
- **Caching**: Second, web servers should cache OCSP responses for efficiency. However, web servers should respect the expiration time of OCSP responses (`nextUpdate`), and remove them from the cache once they have expired.
- **Availability**: Third, when fetching an updated OCSP response, web servers may encounter an OCSP responder that is unavailable or returns an error (e.g., `tryLater` [34]). In such cases, the web server should retain the existing cached OCSP response until it expires (while periodically retrying with the OCSP responder).

Test suite implementation We first implement an OCSP responder that we control by modifying the Python `ocspresponder` library [25]. We also implement our test suite with a Python script that generates a unique DNS name we control, then uses OpenSSL to generate a certificate chain with (1) the OCSP URL that we manage in the AIA extension and (2) the OCSP Must-Staple extension.

Experiment	Apache	Nginx
Prefetch OCSP response	✗ (pause conn.)	✗ (provide no resp.)
Cache OCSP response	✓	✓
Respect `nextUpdate` in cache	✗	✓
Retain OCSP response on error	✗	✓

Table 3: Table showing our experiment results on web servers' correct implementation of OCSP Must-Staple support. Neither popular web server does so completely correctly.

We examine two popular web server software implementations: Apache version 2.4.18 and Nginx version 1.13.12.[26]

Results Our results are summarized in Table 3; we make a few observations below. *First*, we notice that both web servers do not prefetch the OCSP responses even though they serve certificates with the Must-Staple extension; instead, they fetch an OCSP response when they receive the first incoming connection that uses a given certificate. However, their behavior differs with how they respond to this first client: Apache "pauses" the TLS handshake until the OCSP response comes in, while Nginx simply does not provide an OCSP stapled response to the first client. Thus, the first client(s) using Apache will experience delays, while those using Nginx that respect OCSP Must-Staple (e.g., Firefox) will refuse to accept the certificate.[27] *Second*, we observe that Apache does not respect the expiration time of the OCSP response and will continue to serve OCSP responses from the cache even after they expire. In contrast, Nginx will fetch a new OCSP response.[28] *Third*, we observe that when the web server encounters an error communicating with the OCSP responder (unavailability or an error response), Apache also deletes the old (still valid) OCSP response and either provides no OCSP response (if the OCSP responder is unavailable) or serves the error response itself (if the OCSP responder returns an error). In contrast, Nginx retains the old OCSP response and keeps providing it to clients until it expires.

In summary, we observe that both well-known web servers do not fully support OCSP Must-Staple correctly. We believe these undesirable behaviors of web servers could become a serious challenge for web administrators who choose to serve certificates with the OCSP Must-Staple extension: the clients that correctly support the extension might not be able to connect to their domains intermittently due to the above issues. We responsibly reported these issues to Apache Bugzilla [26].

8 CONCLUDING DISCUSSION

We examined whether today's web is ready for OCSP Must-Staple by measuring three major principals—OCSP responders, clients

[26]Note that neither web server implementation supports OCSP Stapling by default; we were required to enable a few configuration parameters.

[27]We noticed that this bug was reported over 3 years ago [12], but it has not fixed yet.

[28]Actually, Nginx does not refresh the cache more than once every 5 minutes; hence, if the validity period of an OCSP response is less than 5 minutes, clients could receive an expired (cached) OCSP response.

(browsers), and web servers—to see if what they are doing correctly supports OCSP Must-Staple.

We observe that 36.8% of OCSP responders experienced at least one outage which typically lasted a few hours. Considering that the OCSP responses can be cached and their median validity periods are a week, we believe that OCSP responders would not be a barrier for the wide adoption of OCSP Must-Staple deployments, although availability of OCSP responders is certainly an area that could be improved. However, the number of certificates that already supports OCSP Must-Staple is minuscule; only 29,709 (0.02%) certificates currently support it.

We develop a test suite for web browsers and deploy it to the most recent versions of all major browsers on both mobile and desktop devices. Overall, we find that the fraction that correctly support OCSP Must-Staple is surprisingly low: all of the browsers other than Firefox do not bother to ensure that stapled OCSP responses are actually included. Lastly, we also checked if popular web server software (Apache and Nginx) correctly fetch and serve OCSP responses. Unfortunately, we find that neither of them prefetch an OCSP response, which introduces unnecessary latency in completing the SSL/TLS handshake with clients. We also found two implementation problems in Apache: it serves expired OCSP responses from the cache and discards previous, valid OCSP responses when it encounters an error communicating with the OCSP responder.

Considering OCSP Must-Staple can operate *only if* each of the principals in the PKI performs correctly, we conclude that, currently, the web is not ready for OCSP Must-Staple.

Recommendations However, on a positive note, we observe that, with correct action by relatively few parties, transition to OCSP Must-Staple would be easy for web site operators. We make the following recommendations to that end:

(1) **Certificate authorities:** Unlike other revocation mechanisms, OCSP Must-Staple is designed to follow a hard-fail approach—clients *must* reject the OCSP Must-Staple enabled certificate if they do not receive a stapled OCSP response from the web server. Therefore, Certificate Authorities (CAs) should bolster the availability and reliability of their OCSP responders.
    We recommend improvements on two broad fronts: First, OCSP responders ought to test the validity of their responses. Test harnesses like ours can help towards this end (we will be making our code and data publicly available). Second, our results show that OCSP responders have variable reachability to different regions of the world. An important area of improvement for OCSP Must-Staple to work is to develop globally reachable and reliable services. The networked systems community has extensive experience in this domain, such as with distributed hash tables [28] and global replication [16]; this would be a logical application of such techniques.
(2) **Web server software developers:** Web server software should pre-fetch OCSP responses from the OCSP responders on a regular basis even if there are no clients who have attempted to make TLS connections. This will help reduce unnecessary latency to clients during their TLS handshakes and cope with intermittent unavailability and errors of OCSP responders. Our results indicate that, for most CAs and most locations, web servers would not have to be very aggressive, as most failures persist far shorter than most OCSP responses' validity periods.
(3) **Browsers:** Clients must begin to actually check if OCSP Staple responses are delivered, fully validate those responses, and hard-fail if a staple is invalid or unavailable. We doubt that browsers will be a driving force behind OCSP Must-Staple; until web servers proactively fetch and OCSP responders deliver valid responses, clients will have little incentive to hard-fail.

Only when *each* of these parties performs these actions can website administrators reliably employ Must-Staple.

9 ACKNOWLEDGMENTS

We thank the anonymous reviewers and our shepherd, Brian Trammell, for their helpful comments. This research was supported in part by NSF grants CNS-1409249, CNS-1563320, and CNS-1564143, and made possible by Akamai Technologies and Cloudflare.

REFERENCES

[1] D. E. 3rd. Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066, IETF, 2011.
[2] C. Arthur. DigiNotar SSL certificate hack amounts to cyberwar, says expert. *The Guardian.* http://www.theguardian.com/technology/2011/sep/05/diginotar-certificate-hack-cyberwar.
[3] J. Amann, O. Gasser, Q. Scheitle, L. Brent, G. Carle, and R. Holz. Mission Accomplished? HTTPS Security after DigiNotar. *IMC*, 2017.
[4] Alexa Top Global Sites. http://www.alexa.com/topsites.
[5] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, IETF, 2008. http://www.ietf.org/rfc/rfc5280.txt.
[6] F. Cangialosi, T. Chung, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson. Measurement and Analysis of Private Key Sharing in the HTTPS Ecosystem. *CCS*, 2016.
[7] T. Chung, Y. Liu, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson. Measuring and Applying Invalid SSL Certificates: The Silent Majority. *IMC*, 2016.
[8] O. Dubuisson. *ASN.1 communication between heterogeneous systems.* Morgan Kaufmann, 2001.
[9] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman. Censys: A Search Engine Backed by Internet-Wide Scanning. *CCS*, 2015.
[10] Z. Durumeric, J. Kasten, D. Adrian, J. A. Halderman, M. Bailey, F. Li, N. Weaver, J. Amann, J. Beekman, M. Payer, and V. Paxson. The Matter of Heartbleed. *IMC*, 2014.
[11] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman. Analysis of the HTTPS Certificate Ecosystem. *IMC*, 2013.
[12] Fetch OCSP responses on startup, and store across restarts. https://trac.nginx.org/nginx/ticket/812.
[13] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman. Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices. *USENIX Security*, 2012.
[14] P. Hallam-Baker. X.509v3 Transport Layer Security (TLS) Feature Extension. RFC 7633, IETF, 2015.
[15] D. Kumar, Z. Wang, M. Hyder, J. Dickinson, G. Beck, D. Adrian, J. Mason, Z. Durumeric, J. A. Halderman, and M. Bailey. Tracking Certificate Misissuance in the Wild. *IEEE S&P*, 2018.

[16] J. Kubiatowicz. OceanStore: an architecture for global-scale persistent storage. *ASPLOS*, 2000.
[17] A. Langley. Revocation checking and Chrome's CRL. 2012. https://www.imperialviolet.org/2012/02/05/crlsets.html.
[18] A. Langley. No, don't enable revocation checking. 2014. https://www.imperialviolet.org/2014/04/19/revchecking.html.
[19] B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. RFC 6962, IETF, 2013. http://www.ietf.org/rfc/rfc6962.txt.
[20] J. Larisch, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson. CRLite: a Scalable System for Pushing all TLS Revocations to Browsers. *IEEE S&P*, 2017.
[21] Y. Liu, H. H. Song, I. Bermudez, A. Mislove, M. Baldi, and A. Tongaonkar. Identifying Personal Information in Internet Traffic. *COSN*, 2015.
[22] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, A. Schulman, and C. Wilson. An End-to-End Measurement of Certificate Revocation in the Web's PKI. *IMC*, 2015.
[23] Let's Encrypt. https://github.com/certbot/certbot/pull/2667.
[24] S. Matsumoto, P. Szalachowski, and A. Perrig. Deployment Challenges in Log-based PKI Enhancements. *EuroSec*, 2015.
[25] OCSP Responder. https://github.com/threema-ch/ocspresponder.
[26] OCSP Stapling should not serve OCSP responses from the cache even after they expire. https://bz.apache.org/bugzilla/show_bug.cgi?id=62400.
[27] Y. Pettersen. The Transport Layer Security (TLS) Multiple Certificate Status Request Extension. RFC 6961 (Proposed Standard), IETF, 2013.
[28] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. *Middleware*, 2001.
[29] M. D. Ryan. Enhanced Certificate Transparency and End-to-End Encrypted Mail. *NDSS*, 2014.
[30] A. Schulman, D. Levin, and N. Spring. RevCast: Fast, Private Certificate Revocation over FM Radio. *CCS*, 2014.
[31] E. Stark, L.-S. Huang, D. Israni, C. Jackson, and D. Boneh. The Case for Prefetching and Prevalidating TLS Server Certificates. *NDSS*, 2012.
[32] N. Sullivan. High-reliability OCSP stapling and why it matters. CloudFlare, 2017. https://blog.cloudflare.com/high-reliability-ocsp-stapling/.
[33] Q. Scheitle, T. Chung, J. Hiller, O. Gasser, J. Naab, R. van Rijswijk-Deij, O. Hohlfeld, R. Holz, D. Choffnes, A. Mislove, and G. Carle. A First Look at Certification Authority Authorization (CAA). *CCR*, 48(2), 2018.
[34] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 6960 (Proposed Standard), IETF, 2013.
[35] E. Topalovic, B. Saeta, L.-S. Huang, C. Jackson, and D. Boneh. Towards Short-Lived Certificates. *W2SP*, 2012.
[36] The Problem with OCSP Stapling and Must Staple and why Certificate Revocation is still broken. https://blog.hboeck.de/archives/886-The-Problem-with-OCSP-Stapling-and-Must-Staple-and-why-Certificate-Revocation-is-still-broken.html.
[37] B. VanderSloot, J. Amann, M. Bernhard, Z. Durumeric, M. Bailey, and J. A. Halderman. Towards a Complete View of the Certificate Ecosystem. *IMC*, 2016.
[38] S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage. When Private Keys Are Public: Results from the 2008 Debian OpenSSL Vulnerability. *IMC*, 2009.
[39] L. Zhu, J. Amann, and J. Heidemann. Measuring the Latency and Pervasiveness of TLS Certificate Revocation. *PAM*, 2016.
[40] L. Zhang, D. Choffnes, T. Dumitraş, D. Levin, A. Mislove, A. Schulman, and C. Wilson. Analysis of SSL certificate reissues and revocations in the wake of Heartbleed. *IMC*, 2014.
[41] L. Zhang, D. Choffnes, T. Dumitraş, D. Levin, A. Mislove, A. Schulman, and C. Wilson. Analysis of SSL Certificate Reissues and Revocations in the Wake of Heartbleed. *CACM*, 61(3), https://cacm.acm.org/magazines/2018/3/225489-analysis-of-ssl-certificate-reissues-and-revocations-in-the-wake-of-heartbleed/fulltext, 2018.