Measuring the Latency and Pervasiveness of TLS Certificate Revocation

Liang Zhu[1] Johanna Amann[2] John Heidemann[1]

[1]USC Information Sciences Institute [2]International Computer Science Institute

Abstract. Today, Transport-Layer Security (TLS) is the bedrock of Internet security for the web and web-derived applications. TLS depends on the X.509 Public Key Infrastructure (PKI) to authenticate endpoint identity. An essential part of a PKI is the ability to quickly revoke certificates, for example, after a key compromise. Today the Online Certificate Status Protocol (OCSP) is the most common way to quickly distribute revocation information. However, prior and current concerns about OCSP latency and privacy raise questions about its use. We examine OCSP using passive network monitoring of live traffic at the Internet uplink of a large research university and verify the results using active scans. Our measurements show that the median latency of OCSP queries is quite good: only 20 ms today, much less than the 291 ms observed in 2012. This improvement is because content delivery networks (CDNs) serve most OCSP traffic today; our measurements show 94% of queries are served by CDNs. We also show that OCSP use is ubiquitous today: it is used by *all* popular web browsers, as well as important non-web applications such as MS-Windows code signing.

1 Introduction

Transport Layer Security (TLS), the successor to Secure Socket Layer (SSL) is one of the key building blocks of today's Internet security. It provides authentication through its underlying X.509 Public Key Infrastructure (PKI) as well as encryption for end-to-end communication over the Internet such as online banking and e-mail.

With the millions of certificates that are part X.509 PKI, it is inevitable that some private keys will be compromised by malicious third parties, lost, or corrupted. An attacker that manages to get access to a certificate's private key can impersonate its owner until the certificate's expiration date. Heartbleed is one example where the private keys of certificates were potentially exposed [9,24]. Even more risky than attacks on individual certificates and keys are attacks on the infrastructure of specific Certificate Authorities (CAs), which can issue certificates for any server (e.g. [7,6,5]).

Two primary mechanisms exist to revoke certificates: Certificate Revocation Lists (CRLs) [8] which provide downloadable lists of revoked certificates, and the Online Certificate Status Protocol (OCSP) [18] which allows clients to check for revoked certificates by sending short HTTP requests to servers of the respective CA. Alternatively, OCSP stapling [17] allows revocation information to be sent by the server in the initial TLS handshake. Some in the security community question the usefulness and viability of these approaches, citing privacy, speed, and other concerns [20,11].

Today, most major web browsers do not reliably check certificate revocation information [12], thus opening their users up to attacks.

In this work, we examine live traffic at the Internet uplink of the University of California at Berkeley (UCB) to check the pervasiveness and latency of OCSP, and then confirm our conclusions with active measurements from two sites.

The primary contribution of this paper is new measurements of OCSP that show that OCSP latency has improved significantly since 2012. We see a median latency of only 20 ms (§ 4), far lower than the 291 ms reported in previous studies [20]. We show that one reason for this improvement is that most OCSP traffic today is served by content delivery networks (CDNs). Our second contribution is a cost evaluation of OCSP connections. We identify that OCSP verification typically accounts for 10% of the TLS setup time. OCSP will almost never delay TLS when being run in parallel with the TLS handshake, and it only adds a modest delay if run sequentially (§ 4.3). Our final contribution is examination of how OCSP is being used today: *all* popular web browsers and important non-web applications such as MS-Windows code signing (§ 3) use OCSP. Furthermore, 88% of the IPv4 addresses that perform TLS queries during our measurement also perform OCSP queries.

2 Data Collection

Our study uses passive data collected from live Internet traffic to determine characteristics and features of OCSP use. We augment our passive data with information from active scans to verify our timing results and to check which OCSP servers use CDNs. We use passive measurements to study how OCSP is actually used on the Internet, and to evaluate the interplay between server and client software. These passive measurements are from a specific site (UCB), so our passive results depend on what sites that population visits. We take active probes from two sites, Berkeley and the University of Southern California. While this data source may bias our results, Berkeley has a large user population and we probe many observed sites, so our data reflects the real experiences of this population, and does not reflect outliers due to rarely used servers. Our active measurements are from two sites (to avoid some bias), but both are well connected and users with slower connectivity may experience higher latencies. We believe this dataset is informative and reflects the lookup performance of current OCSP servers and their use of CDNs, even if future work is needed to confirm the results from other viewpoints. These risks are common to many measurement studies that depend on data from large, real populations where multiple data sources are difficult to obtain due to privacy concerns around network measurement.

For our data collection, we extended the Bro Network Monitor [2,15] with the capability to parse OCSP requests and responses. Bro uses a file signature (expressed as a regular expression) to detect OCSP requests and replies. We correlate OCSP messages with TLS connections using IP addresses, certificate hashes and timing information (see § 4.3). Our changes will be integrated in the next version of Bro.

category		**application**	**percent**
Web browsers	32.10%	Firefox	31.63%
		Chrome	.21%
		Pale moon	.06%
		Opera	.06%
		Rekonq, Bolt, Midori, Iceweasel, Seamonkey, Safari Sonkeror, IE, Camino, Epiphany, Konqueror	<.15%
Library or daemon used by applications	66.87%	ocspd	37.15%
		Microsoft-CryptoAPI	23.74%
		securityd	4.74%
		java	1.24%
		cfnetwork	<.0001%
Email client	.32%	Thunderbird	.30%
		Postbox, Gomeza, Zdesktop, Eudora, Icedove	.02%
Other applications	.33%	Lightning	.31%
		Zotero	.01%
		Celtx, ppkhandler, Komodo, Dalvik, slimerjs, Unity Phoenix, Sunbird, Slurp, miniupnpc, googlebot Entrust entelligence security provider	<.0074%
Unknown	.38%	Unknown	.38%

Table 1: OCSP applications (based on HTTP user agent) observed in 41.87 M OCSP HTTP requests. Date: 2015-07-28 to 2015-09-28.

Our passive measurements cover 56 days of data taken between 2015-07-28 to 2015-09-28 at the Internet uplink of the University of California at Berkeley (UCB). We record data for only 56 days of this 63-day period because of outages due to hardware failures, fire, and preemption by another study that required complete access to the hardware for about a week. We observed 1690 M TLS connections with certificates encountered and about 42 M OCSP requests over this period.

After processing the data we noticed that in 0.43% of the OCSP connections, we have zero (or in a handful of cases negative) lookup times. We verified the correctness of our measurement manually against network traces and were not able to reproduce these error cases. We believe these impossible results are caused by interactions between packet retransmissions and Bro.

3 OCSP Use in Applications and Hosts

We first want to understand how widely OCSP is used—how many applications and hosts make OCSP queries.

Applications: We evaluate which applications use OCSP by examining the user-agent header of the OCSP requests. Table 1 shows the resulting distribution of user-agents. The majority of the lookups are done by Firefox and system libraries and daemons: Microsoft-CryptoAPI (Windows) and ocspd (Mac OS).

To understand this distribution, we examine the behavior of common Internet Browsers (IE, Chrome, Firefox, Safari) and operating systems. We find that Firefox always uses its own user-agent, which is attributable to the fact that it uses its own encryption library [1]. Microsoft Internet Explorer and Safari use their respective operating system functionality for OCSP lookups, not directly revealing their user-agents. Google Chrome only uses OCSP for extended Validation certificates [11,12]. It uses the operating system functionality for OCSP lookups on Windows and Mac OS. On Linux, it performs OCSP requests with its own user-agent.

This use of libraries makes it difficult to distinguish the different browsers. This problem is exacerbated by the fact that a manual examination of OCSP requests revealed that Windows and Mac OS also perform OCSP requests for application signatures with the same user-agent. When we examine all unique OCSP requests (those for different certificates), we see that 81% of these unique certificates account for nearly all (95%) of total OCSP requests observed on the wire. Hence, the number of code-signing requests is at most the number of requests without matching certificates in traffic: 5% of all OCSP requests and at 19% of the unique requests encountered.

Application Comments: While examining the OCSP requests, we noticed a number of software bugs in different implementations. According to the respective standard, an OCSP request sent with HTTP GET will be base64 and then URL-encoded. Some clients do not adhere to this standard, skipping the URL-encoding of requests. Servers still seem to accept these malformed requests. In our dataset, 99.9% of these non-standard requests were caused by the Apple ocspd versions 1.0.1 and 1.0.2. The bug was apparently fixed in version 1.0.3, appearing in MacOS 10.10. We also encountered requests where the user-agent only contains the string representation of a memory address.

Clients can choose which hash algorithm they wish to use in an OCSP requests. During our monitoring effort, all clients used SHA1.

During a random day (2015-08-24), the median size of the OCSP requests and responses were 300 and 1900 bytes.

Use of OCSP by Hosts: To evaluate how many hosts send OCSP, we examine how many IP addresses send both OCSP and TLS traffic. We found that *88% of IPv4 addresses using TLS also send OCSP*, suggesting widespread use of OCSP. We do not measure IPv6 addresses because hosts exchange web traffic via TLS on IPv6 but issue their OCSP request via IPv4. Underuse of IPv6 for OCSP is likely because of limited support of IPv6 in OCSP servers: only 45% of the 304 unique OCSP servers we observe have an IPv6 address.

Please note that Network Address Translation may cause an overestimate of OCSP deployment. Ideally, one would want to determine the exact number of connections that use OCSP; however performing such measurements would require simulating the use of OCSP caching and is beyond the scope of this paper.

4 Latency of OCSP

Web browsing is very sensitive to latency, and there have been concerns that the latency introduced by OCSP is too high [11]. In this section, we study OCSP latency in three ways. First, we measure OCSP latency in live Internet traffic in § 4.1. Then, we verify these results with active probes of OCSP servers in § 4.2. Finally, we compare OCSP latency to the TLS connection setup latency in § 4.3.

4.1 OCSP Delay in Network Traffic

As a first step, we use our passive dataset (§ 2), to analyze the distribution of OCSP latency.

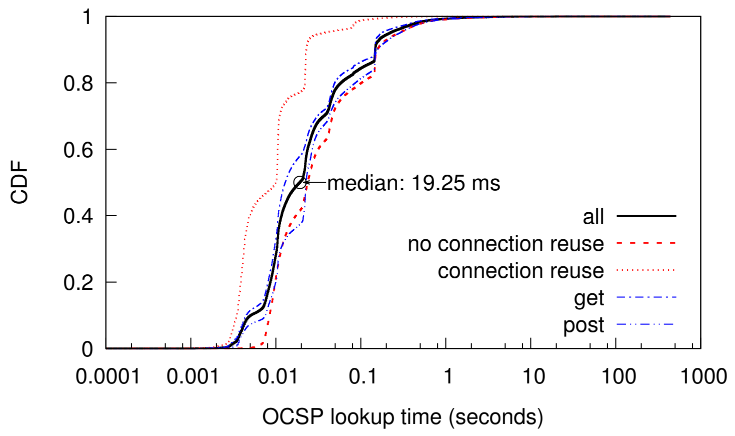


Fig. 1: Cumulative distribution of OCSP lookup time including the TCP handshake time for the first OCSP request in HTTP connection, over 41.12 M OCSP lookups. Date: 2015-07-28 to 2015-09-28

Methodology: We define *OCSP lookup time* as the time from setting up a new TCP connection to getting the first OCSP response. When multiple OCSP responses are pipelined over a single TCP connection, we define the lookup time for subsequent requests from start of request to the end of the corresponding response. This definition reflects the amortization of connection setup time over several requests, but it may underrepresent the user-perceived time if requests arrived in a burst.

CDNs used in traffic: We find that overall the *current* OCSP lookup is very quick with a median time of 19.25 ms (Figure 1). Even when we include the connection setup times by considering only new HTTP connections, the median OCSP lookup time is still only 23.78 ms. Studies by Stark et al. in 2012 showed medians 14× larger [20] (291 ms compared to our 19 ms). Although most lookups are fast, the distribution of times has a long tail, with a very few (less than 0.1%) taking 5 seconds to 8 minutes.

We believe the primary reason OCSP performance has improved since 2012 is that today most OCSP traffic is served by CDNs. To identify OCSP queries going to CDNs we mapped IP addresses in the traffic to hostnames and well known CDNs (like Akamai, Edgecast, and Google) or the presence of `CDNs` in the reverse hostname.

Table 2 shows the fraction of lookups (dynamic traffic) and servers (static OCSP sites) that we identify as being served or hosted by CDNs. While only 39% of the servers that are accessed in our passive measurements are hosted by known CDNs, we see that these servers manage the popular certificates: *more than nine-tenths of queries* (94%) are served by CDNs. Service is quite heavily skewed, with the 68% of traffic serviced by the top 10 busiest OCSP servers (Table 3). All of them are handled by third party or internal CDNs.

	Query Traffic		OCSP Servers	
CDN	39313464	94%	120	39%
other	2526338	6%	184	61%
total	41839802	100%	304	100%

Table 2: CDN usage of 304 unique OCSP servers discovered in our passive monitoring over two months. Date: 2015-07-28 to 2015-09-28

server	**observed CDN**	**lookup**	
ocsp.digicert.com	phicdn.net	6,205,125	14.83%
clients1.google.com	self-hosted	4,859,409	11.61%
sr.symcd.com	akamaiedge	3,778,672	9.03%
ocsp.entrust.net	akamaiedge	2,421,420	5.79%
ocsp.godaddy.com	self-hosted (using akadns)	2,399,931	5.74%
ocsp.usertrust.com	self-hosted	2,248,577	5.37%
vassg141.ocsp.omniroot.com	akamai	1,915,287	4.58%
ss.symcd.com	akamaiedge	1,663,053	3.97%
ocsp.comodoca.com	self-hosted	1,478,911	3.53%
ocsp.verisign.com	akamaiedge	1,345,724	3.22%
all 294 others		13,523,693	32.32%
total		41,839,802	100%

Table 3: Top 10 busy OCSP servers and their lookups discovered in our passive monitoring. Date: 2015-07-28 to 2015-09-28

CDNs seen on servers: To get further evidence of the use of CDNs by CAs, we examine the certificates of an Internet-wide scan of TCP port 443 by Rapid7 Labs [3]. Using their scan of 2015-09-28, we extract a list of 455 unique OCSP servers. This list includes 57% of the OCSP servers we discovered, but neither list subsumes the other. We evaluate this list for CDNs using the same method as before. We find that 29% of the OCSP servers are invalid (non-existent domain), which is probably cased by misconfigurations, outdated, or internal certificates. Of all certificates with valid servers, 23% are served by CDNs, confirming that many CAs use CDNs for their OCSP servers. It also shows that CDN use is more common in certificates of popularly used servers than in all certificates. We believe this skew to be caused by the fact that popular services keep their certificates updated better than the "average" TLS user. This result again shows the importance of studying dynamic traffic to differentiate typical OCSP performance from the worst case.

We have two additional observations about OCSP latency. First, we see that `GET` requests are faster than `POST` requests (median 13.0 ms compared to 22.8 ms, Figure 1). The HTTP standards recommend GET for short requests, and we see about half of all OCSP requests using this method.

Finally, we see that it is not uncommon for OCSP requests to reuse an existing HTTP connection, avoiding connection setup latency. In our measurements, 24% of all OCSP lookups reuse a connection. Examining random samples of OCSP requests that were reused reveals that connection reuse has several likely causes: webpages that include resources from several other pages that share the same OCSP servers, users accessing pages that share the same OCSP server quickly to each other, and checks for end-host and intermediate certificates that share the same OCSP server. Connection reuse reduces the overhead significantly: OCSP queries that reuse connections complete with a median of 10 ms; less than half that of those that start new connections (24 ms).

Our data includes OCSP requests for both intermediate certificates and leaf certificates. Our analysis reflects the overall lookup performance of OCSP servers; we do not study specific types of certificates.

4.2 OCSP Server Delay

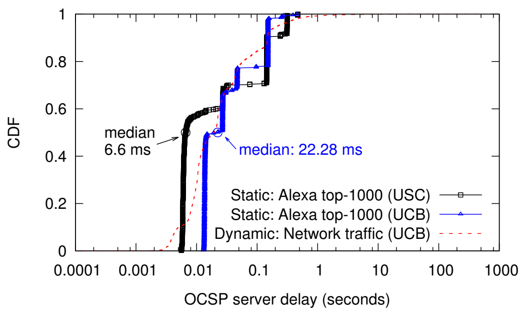
Our passive study of OCSP traffic emphasizes the performance of the most commonly used servers. We next augment our study with observations of active probes to OCSP servers, to verify the results of our passive measurements and capturing a static picture of the time an application takes to verify the validity of certificates.

Methodology: We actively probe OCSP servers of the Alexa top-1000 from two different vantage points, UCB and USC. We perform an HTTPS connection attempt for each site (Figure 2a). We discard 362 (USC: 364) sites with failing DNS lookups, where servers not answer to HTTPS requests or where we cannot obtain valid certificate chains. We obtain complete certificate chains for the remaining 638 (USC: 636) sites. We identify 508 (USC: 506) unique end host certificates, discarding 130 (USC: 130) duplicate certificates (typically by sites operated by the same company, such as `youtube.com` and `google.com`). We then query the OCSP servers to check each end certificate using a custom program that employs the `OpenSSL` library to send OCSP requests via HTTP POST. We record the query start and response times. We conducted this experiment on two well connected, capable machines (32-core with x86-64 Fedora 21 Linux 4.0.5 and 4-core with x86-64 Fedora 22 Linux 4.2.6). We repeat each query 20 times and report the median value to avoid outliers.

Our active probes show overall short latencies with a median of 22.28 ms at UCB (Figure 2b), which is similar to the median of OCSP network delay measured by passively collected data (§ 4.1). It also shows that computational cost for generating OCSP request and parsing response is small; in our experiment, the time to generate an OCSP request is normally less than 0.5 ms. The latency of most OCSP requests is acceptable: at UCB, 77% of the OCSP queries are completed within 50 ms, although there are also some tardy responses (22%) taking more than 150 ms. This also confirms our passive measurements of network delay and reinforces that lookup time improved significantly compared to [20]. Our measurements from USC show a similar distribution of OCSP lookup performance, but with slightly smaller latency (median 6.6 ms). We think the difference is caused by fewer hops to CDNs from our vantage point at USC. The stepped pattern in Figure 2b is caused by certificates sharing the same OCSP servers and the speed of the different CDNs.

sites	**UCB**	**USC**
considered	1000	1000
no IPv4	29	27
no TLS	308	310
TLS	663	663
no cert/chain	25	27
duplicates	130	130
unique certs	508	506
no ocsp url	2	2
complete	506	504

(a) Certificates retrieved.



(b) Cumulative distribution of OCSP delay.

Fig. 2: Evaluating OCSP across the Alexa top-1000 websites. Date: 2016-01-09.

4.3 OCSP Overhead in TLS

Our measurements show only modest OCSP delays. However, this cost needs to be put into the context of overhead it adds to the TLS connection setup. We now examine how OCSP affects TLS performance during session establishment, using our passive dataset (§ 2). We define *TLS delay* as the time between the *client hello* message and the first encrypted application data packet sent by client. During an OCSP query, the TLS handshake can either be interrupted until an OCSP response is received, or continue in parallel. In the parallel case, the client must not send its first request to the server until receiving a valid OCSP response.

Matching OCSP requests to TLS connections: To understand the overhead OCSP adds to TLS, we must map OCSP messages transmitted via HTTP to their corresponding TLS connections. We log all TLS connections and information about their certificates in addition to all OCSP requests and responses. We then match OCSP requests to TLS connections using the 4-tuple (*source ip, ocsp URL, issuer name hash, serial number*) from both flows and identify the TLS connection closest in time to the OCSP request. We identify and discard cases where the OCSP request precedes the TLS connection (an *early request*), and when it follows by more than 10 s (*a late request*).

Using the method above, we successfully correlate 52% of the 41 M OCSP requests with TLS connections (*matched requests*). We discard 17% as early requests, 1.8% as late requests and are unable to match 30% using the 4-tuple (*unmatched requests*).

Although we match the majority of requests, the high mismatch rate (including impossible early requests) stems from several challenges in matching. We believe a large number of mismatches are caused by dual-stack, IPv4/v6 hosts where TLS connections occur on IPv6 but where the OCSP servers only support IPv4. While 88% of IPv4 addresses send both TLS and OCSP requests, 90% of IPv6 addresses send no OCSP requests. OCSP requests caused by non-TLS services, such as code-signing [22] are another reason for unmatched OCSP requests [22] (see § 3) Finally, while the reported packet-loss in our monitoring infrastructure

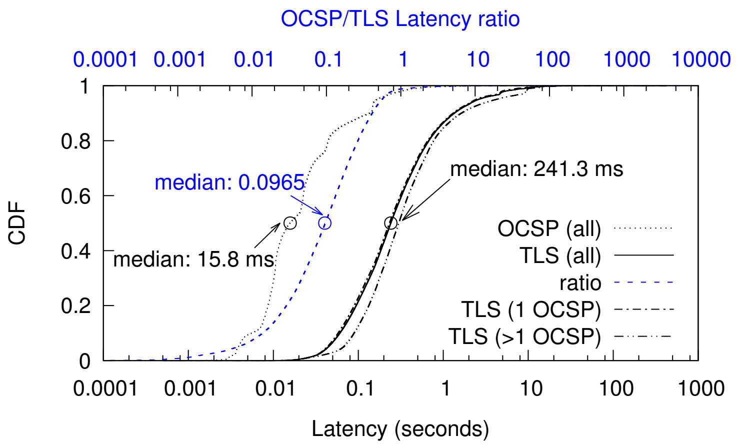


Fig. 3: Cumulative distribution of OCSP network latency and TLS delay for all matched pairs. Date: 2015-07-28 to 2015-09-28. The blue dotted line shows the cumulative distribution of the ratio of the sum of OCSP network latency to TLS delay for every matched pair.

is low (about 1% of packets), it may still prevent the identification and parsing of some TLS and OCSP connections. To avoid errors, we use only matched requests in the following analysis.

Finally, we filter empty TLS and OCSP queries: we discard 11% of TLS connections that have no client application data and 0.9% of OCSP lookups that are missing either a request or response.

OCSP lookup in TLS Delay: Using the paired OCSP queries and TLS connections, we evaluate how much latency the OCSP lookup adds to TLS connections in Figure 3.

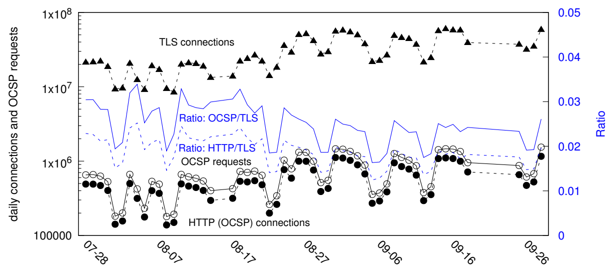
The key result is that *OCSP typically accounts for one-tenth of the total TLS delay.* We see a median TLS delay of 242 ms compared to a median OCSP lookup time of 15.8 ms. We compute the ratio of OCSP lookup time to TLS delay for each paired connection with a median of 0.0965. We see some outliers (1.2%) where the OCSP lookup time exceeds the TLS delay; we expect these cases to be caused by timeouts.

The actual delay OCSP incurs to the user depends on the structure of the application. Many applications do OCSP validation in parallel with starting the TLS connection. Our evaluation shows that OCSP lookup time is only about 10% of TLS delay overall, as shown by the median latency ratio, and the cost is typically 16 ms. This cost suggests that, *if OCSP is performed in parallel with TLS session setup, the OCSP delay is almost never visible.*

In summary, the OCSP lookup latencies we observe have improved significantly compared to prior reports [20]. OCSP lookup only adds modest delay to TLS setup, and potentially never adds latency when performed in parallel.

validity	**percent**
< 1 day	2%
1–6 days	38%
7–10 days	57%
> 10 days	3%

(a) Distribution of OCSP validity times for 290 k unique certificates.



(b) Daily number of TLS connections, OCSP requests and HTTP (OCSP) connections.

Fig. 4: Evaluating OCSP caching. Date: 2015-07-28 to 2015-09-28.

4.4 Effectiveness of OCSP Caching

A final component of OCSP latency is caching of OCSP responses. Our data does not provide an exact picture of caching, but we can use it to estimate the effectiveness of caching.

The potential of OCSP caching can be seen in the OCSP validity periods. Our passive dataset of OCSP traffic, Figure 4a shows that most OCSP responses have a validity period of a week or more. 95% are valid for at least one day.

To give some estimate of the effectiveness of caching we counted the aggregate number of OCSP requests relative to TLS connections. Figure 4b shows the number of TLS connections and OCSP requests per day, with a mean of 30 M TLS connections and only 0.7 M OCSP requests per day. Since we have shown that most browsers and most IPv4 addresses use OCSP (§ 3), this ratio of 1 OCSP request for 40 TLS connections suggests very effective caching. To understand the exact impact of OCSP caching, future work must distinguish a cache hit from cases where browsers disable OCSP, and from TLS sessions are established by software that does not use TLS.

The potential of long-term OCSP caching is important because it significantly attenuates the information about end-user browsing that is visible to CAs. Since OCSP replies can be cached for at least one day, the information visible over this channel is quite limited.

5 OCSP In Action: Revoked certificates

The point of OCSP is to revoke certificate that are no longer suitable for use, a condition that we expect to be very rare but still very important. OCSP is effective in practice—*we see a few examples of revoked certificates in our data.*

As expected, there are relatively few revoked certificates. We see OCSP replies for 2,180 unique revoked certificates in our passive dataset that contains OCSP replies for 1,418,315 unique certificates. Only 0.3% of OCSP queries report a revoked certificate.

We manually examined the top 10 revoked certificates by number of OCSP requests to understand their use. Seven of these were expired code-signing certificates for software on the Windows platform. The rest were for subdomains of `t-mobile.com`, `aol.com` and `lijit.com` that were inaccessible in October 2015. We speculate that these revocations indicate deployed software that has not been updated and is trying to use discontinued services.

Finally, we observe a very few 638 (0.001%) OCSP responses for 105 unique certificates with the status of "unknown". Searching for cases where the same OCSP responses also returned a different status revealed the cause for 72 of these requests for 12 unique certificates: The most common cause is certificates that have just been issued and are not known to the revocation server yet. For 4 certificates, the CA returned an unknown status, apparently without reason (certificate valid, later replies indicate "good" again). For 1 code-signing certificate, the CA apparently returns unknown after the certificate expired. For the remainder of the requests we could not identify a reason.

6 Related Work

There has been a wealth of work to measure different parts of the TLS and certificate ecosystem, including studies of details of the CA ecosystem [10], TLS errors [4] and certificates contained in root stores [16].

Prior work examined different aspects of TLS certificate revocation. After the 2008 Debian OpenSSL vulnerability and the Heartbleed bug, researchers studied the number of revocations, revocation patterns and patching behavior [23,9,24]. In difference to these studies which focus on certificate revocation patterns after a vulnerability, we study the performance impact of revocation in general. Researchers also proposed to use alternative approaches to certificate revocation like FM radio broadcasts for certificate revocation [19] as well as using short-lived certificates to make revocations unnecessary [21].

Most recently, Liu et al. use full IPv4 scans and compare them with blacklists [12]. They also study revocation checking behavior of web browsers and operating systems as well as Google's certificate revocation infrastructure. In difference to us, they do not study the actual use of OCSP on the Internet or its latency impact on the Internet.

Most related to our work, Stark et al. measured OCSP lookup latency [20]. Like their work, we use active and passive approach to understand OCSP latency. However, we collect network traffic at a university network with a broader coverage. Our data has a more diverse and much larger set of clients. Furthermore, we also compare the speed of OCSP connections to the remainder of the TLS handshake. Netcraft published OCSP performance surveys of major CAs [14,13]. They use static sites to study OCSP latency and reliability. In contrast, our analysis uses live network traffic to understand current OCSP latency.

OCSP stapling [17] was proposed as an alternative. Examining the usage of OCSP stapling and its overhead is future work.

To the best of our knowledge, no previous work examined OCSP network traffic. Our analysis of actual traffic patterns provides insight into dynamic *traffic*, complementing these prior studies that focused on analysis of static sites.

7 Conclusion

Our measurements show that the speed of OCSP servers has increased tremendously. Due to the widespread use of CDNs OCSP almost never has any user-perceived performance cost when done in parallel with TLS setup, and adds only about 10% additional latency if done sequentially (§ 4.1).

Privacy has been a second concern about OCSP—CAs running the OCSP servers can potentially deduce parts of a users browsing behavior. We have shown that OCSP caching means that queries most queries are sent weekly or at most daily, limiting this channel (§ 4.4).

A third concern about OCSP are problems with captive portals—web-pages that require a user to agree to terms and conditions before being able to use the Internet; some of these captive portals use HTTPS. In these cases, the OCSP servers cannot be contacted to verify that the site certificate has not yet been revoked. We leave addressing this problem to future work. One possible approach is to use OCSP stapling [17] for captive portals—in these cases, OCSP lookups would not be necessary. Alternatively, captive portals could allow HTTP connections to specific OCSP servers.

Finally, we have shown that while certificate revocations are quite rare (as expected), they do occur in practice (§ 5).

Ultimately, the data in our paper suggests that OCSP today is both important and viable—it adds minimal or no user-visible delay or privacy, and it provides an essential protection against certificate compromise.

Acknowledgments

This work was supported by the National Science Foundation (NSF) under grant numbers CNS-1528156 and ACI-1348077, by the Department of Homeland Security (DHS) Science and Technology Directorate, HSARPA, Cyber Security Division, via SPAWAR Systems Center Pacific (contract N66001-13-C-3001), and via BAA 11-01-RIKA and Air Force Research Laboratory, Information Directorate (agreements FA8750-12-2-0344 and FA8750-15-2-0224). The U.S. Government is authorized to make reprints for governmental purposes notwithstanding any copyright. The views contained herein are those of the authors and do not necessarily represent those of NSF, DHS or the U.S. Government.

References

1. Network Security Services. `https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS`.
2. The Bro Network Security Monitor. `https://www.bro.org`.
3. Project Sonar: IPv4 SSL Certificates. `https://scans.io/study/sonar.ssl`, Aug. 2015.
4. D. Akhawe, J. Amann, M. Vallentin, and R. Sommer. Here's My Cert, So Trust Me, Maybe? Understanding TLS Errors on the Web. In *WWW*, May 2013.

5. C. Arthur. DigiNotar SSL certificate hack amounts to cyberwar, says expert. http://www.theguardian.com/technology/2011/sep/05/diginotar-certificate-hack-cyberwar, Sept. 2011.
6. S. Bhat. Gmail Users in Iran Hit by MITM Attacks. http://techie-buzz.com/tech-news/gmail-iran-hit-mitm.html, Aug. 2011.
7. Comodo. Comodo Fraud Incident. https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html, Mar. 2011.
8. 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, May 2008.
9. 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. In *ACM IMC*, 2014.
10. R. Holz, L. Braun, N. Kammenhuber, and G. Carle. The SSL Landscape: A Thorough Analysis of the X.509 PKI Using Active and Passive Measurements. In *ACM SIGCOMM*, 2011.
11. A. Langley. Revocation checking and Chrome's CRL. https://www.imperialviolet.org/2012/02/05/crlsets.html, Feb. 2012.
12. Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, and C. Wilson. An End-to-End Measurement of Certificate Revocation in the Web's PKI. In *ACM IMC*, 2015.
13. Netcraft. Certificate revocation and the performance of OCSP. http://news.netcraft.com/archives/2013/04/16/certificate-revocation-and-the-performance-of-ocsp.html.
14. Netcraft. OCSP Server Performance in April 2013. http://news.netcraft.com/archives/2013/05/23/ocsp-server-performance-in-april-2013.html.
15. V. Paxson. Bro: A system for detecting network intruders in real-time. *Comput. Netw.*, 31(23-24):2435–2463, Dec. 1999.
16. H. Perl, S. Fahl, and M. Smith. You Wont Be Needing These Any More: On Removing Unused Certificates from Trust Stores. In *FC*, 2014.
17. Y. Pettersen. The Transport Layer Security (TLS) Multiple Certificate Status Request Extension. RFC 6961, 2013.
18. 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, June 2013.
19. A. Schulman, D. Levin, and N. Spring. RevCast: Fast, Private Certificate Revocation over FM Radio. In *ACM CCS*, 2014.
20. E. Stark, L.-S. Huang, D. Israni, C. Jackson, and D. Boneh. The Case for Prefetching and Prevalidating TLS Server Certificates. In *NDSS*, 2012.
21. E. Topalovic, B. Saeta, L.-S. Huang, C. Jackson, and D. Boneh. Towards Short-Lived Certificates. In *W2SP*, 2012.
22. Wikipedia. Code signing. https://en.wikipedia.org/wiki/Code_signing.
23. S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage. When Private Keys Are Public: Results from the 2008 Debian OpenSSL Vulnerability. In *ACM IMC*, 2009.
24. L. Zhang, D. Choffnes, D. Levin, T. Dumitras, A. Mislove, A. Schulman, and C. Wilson. Analysis of SSL Certificate Reissues and Revocations in the Wake of Heartbleed. In *ACM IMC*, 2014.