# Certificate Transparency in the Wild: Exploring the Reliability of Monitors

Bingyu Li[1,2,3], Jingqiang Lin[1,2,3*], Fengjun Li[4], Qiongxiao Wang[1,2], Qi Li[5], Jiwu Jing[6], Congli Wang[1,2,3]

1. State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences
2. Data Assurance and Communication Security Center, Chinese Academy of Sciences
3. School of Cyber Security, University of Chinese Academy of Sciences
4. Department of Electrical Engineering and Computer Science, the University of Kansas, USA
5. Institute for Network Sciences and Cyberspace, Tsinghua University, China
6. School of Computer Science and Technology, University of Chinese Academy of Sciences

{libingyu, linjingqiang}@iie.ac.cn, fli@ku.edu, wangqiongxiao@iie.ac.cn, qli01@tsinghua.edu.cn, jing@is.ac.cn, wangcongli@iie.ac.cn

## ABSTRACT

To detect fraudulent TLS server certificates and improve the accountability of certification authorities (CAs), certificate transparency (CT) is proposed to record certificates in publicly-visible logs, from which the monitors fetch all certificates and watch for suspicious ones. However, if the monitors, either domain owners themselves or third-party services, fail to return a complete set of certificates issued for a domain of interest, potentially fraudulent certificates may not be detected and then the CT framework becomes less reliable. This paper presents the first systematic study on CT monitors. We analyze the data in 88 public logs and the services of 5 active third-party monitors regarding 3,000,431 certificates of 6,000 selected Alexa Top-1M websites. We find that although CT allows ordinary domain owners to act as monitors, it is impractical for them to perform reliable processing by themselves, due to the rapidly increasing volume of certificates in public logs (e.g., on average 5 million records or 28.29 GB daily for the minimal set of logs that need to be monitored). Moreover, our study discloses that (*a*) none of the third-party monitors guarantees to return the complete set of certificates for a domain, and (*b*) for some domains, even the union of the certificates returned by the five third-party monitors can probably be incomplete. As a result, the certificates accepted by CT-enabled browsers are not absolutely visible to the claimed domain owners, even when CT is adopted with well-functioning logs. The risk of invisible fraudulent certificates in public logs raises doubts on the reliability of CT in practice.

## CCS CONCEPTS

• **Security and privacy** → **Security services**; **Network security**.

## KEYWORDS

Certificate Transparency; Trust Management

**ACM Reference Format:**
Bingyu Li, Jingqiang Lin, Fengjun Li, Qiongxiao Wang, Qi Li, Jiwu Jing, and Congli Wang. 2019. Certificate Transparency in the Wild: Exploring the Reliability of Monitors. In *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS'19), November 11–15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3319535.3345653

## 1 INTRODUCTION

Traditional X.509 public key infrastructures (PKIs) assume trusted certification authorities (CAs) that are responsible for issuing certificates [14]. However, several security incidents indicate that accredited CAs may be compromised or deceived to issue fraudulent TLS server certificates [13, 19, 27, 38, 53, 75, 77, 79], which bind a domain name (e.g., www.facebook.com or www.gmail.com) to a key pair held by man-in-the-middle (MitM) or impersonation attackers, instead of the legitimate website.

Several approaches attempt to tame the absolute authority of CAs from different perspectives [22, 40, 43, 72, 76], but none of them is widely deployed [6]. Certificate transparency (CT) is proposed to detect fraudulent certificates and improve the accountability of CAs [28, 46]. It has been supported by browsers and TLS software, including Chrome [29], Apple platforms [8], Firefox [55], OpenSSL [59], Nginx [56], Microsoft AD Certificate Service and Azure Key Vault [51].

In the CT framework, a certificate is submitted to multiple public servers called *logs* by the CA that issues it or sometimes by the domain owner (or website) for which it is issued. In response, the log generates a signed certificate timestamp (SCT). In TLS negotiations, the server certificate is delivered along with SCTs; otherwise, it will be rejected by CT-enabled browsers. CT ensures that any certificate acceptable to CT-enabled browsers is recorded in publicly-visible logs, so that it is visible to *monitors* for further checks.

It is worth noting that CT does not prevent a CA from issuing fraudulent certificates. CT logs only record all certificates submitted to them and sign the corresponding SCTs, without checking

*Jingqiang Lin is the corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
*CCS '19, November 11–15, 2019, London, United Kingdom*
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6747-9/19/11…$15.00
https://doi.org/10.1145/3319535.3345653

whether a certificate is issued with the domain owner's authorization. Hence, a fraudulent certificate is still acceptable to CT-enabled browsers after being submitted to the logs by the attacker (e.g., a compromised CA). So the CT framework relies on the monitors to retrieve *all* the certificates belonging to the inquired domain in a timely and reliably means to assist the detection of fraudulent certificates. If a fraudulent certificate is missing in the search result returned to users, the attacker could exploit it to launch MitM or impersonation attacks, without triggering any alert in CT. The longer the fraudulent certificates stay undetected in the system (or CT logs), the more the damage they may cause to the PKI ecosystem. Thus, the quality of the search results provided by the CT monitors, especially the *completeness* of the results, affects the overall security enhancement by the CT framework.

However, there is little study in the literature about the *reliability* of the services provided by CT monitors. In practice, many third-party monitors claim to reliably return all known, unexpired certificates for a domain [24, 35, 63], but none of them provides any mechanism about the completeness of the returned results. This work is among the first to study the reliability of CT monitors. We expect a *reliable* monitor to return the *complete* set of certificates for any inquired domain name. With the assistance of such reliable monitors, a domain owner can quickly identify any suspicious certificates issued for its domain. However, this requires the monitor to monitor a large number of, if not all, public CT logs. In practice, it needs to fetch all certificates at least in the default logs that are pre-included in CT-enabled browsers. Meanwhile, the monitor should process the fetched certificates properly to produce the correct answer for each inquiry in the certificate search services.

While the monitors are essential to CT, it remains unclear if the monitors in the wild provide reliable services. A domain owner is allowed to act as a monitor to watch for certificates related to its domain name [28, 44]. Meanwhile, there are five active *third-party monitors*,[1] crt.sh [12], SSLMate [62], Censys [73], Google Monitor [31] and Facebook Monitor [23], providing certificate search services. They fetch certificates from the public logs and return certificates related to the inquired domain name.

This paper presents the first systematic study on CT monitors. We analyzed the certificates in 88 public logs and the certificate search services of 5 active third-party monitors regarding 3,000,431 certificates of 6,000 selected Alexa Top-1M websites [4]. All data were collected and all certificate searches were conducted on October 27, 2018, except two experiments for ordinary domains in January 2019. Our study uncovers several problems on the implementation and deployment of CT monitors. First, *acting as a monitor raises storage and network bandwidth requirements that are beyond the capacity of most ordinary domain owners.* By October 2018, there are over 2.87 billion certificates in 88 public logs, which consume about 15.86 TB of storage space. Among them, 50 logs servers that accept the certificates trusted by common TLS clients and serve normally (referred as *regular logs* in this paper) maintain 2.77 billion records at a size of 15.31 TB. Moreover, the number is increasing dramatically, at an average rate of 6,542,421 records per day in 88 logs and 6,275,652 per day in 50 regular logs in 2018. It is extremely costly for an ordinary domain owner to act as the monitor by itself, to fetch and process the rapidly increasing volume of certificates (about 30 GB per day at least) in the public logs.

Moreover, our study of the third-party monitor services shows that *none of the third-party monitors guarantees to return the complete set of valid certificates recorded in the public logs for a domain.* We studied two groups of domains, one for the popular Alexa Top-1K websites and the other for less popular ones (5,000 domains randomly selected from Alexa Top-1M websites), and searched certificates for all these domains from 5 third-party monitors. In both cases, none of the monitors returns the complete sets of certificates for all the inquired domain names, which also supports our first finding that an ordinary domain owner is less capable of acting as a monitor to process all records in the logs. The incompleteness of the returned search results may cause some fraudulent certificates in public logs to be invisible to the claimed domain owners and thus evade the detection, which makes the CT framework unreliable.

To the best of our knowledge, this is the first work to analyze CT monitors in the wild. Existing studies on CT [6, 37, 58, 67, 70, 74] focus on the deployment of log servers and the adoption in websites or browsers. While providing different views of CT, these large-scale studies do not investigate the reliability of monitors in practice. Our study identifies challenges in the implementation of CT monitors and discloses the vulnerabilities in third-party monitors.

In summary, the contributions of this paper are as follow: (a) We perform the first systematic study of the reliability of monitors for the purpose of studying the effectiveness of CT. (b) We investigate various types of log sets and find that each monitor can monitor a minimal set of logs while ensuring the reliability. (c) We measure the reliability of all mainstream third-party monitors and evaluate the completeness of certificates returned by their certificate search services. (d) We analyze the possible causes of incomplete certificate search results and discuss several improvements to enhance the reliability of monitors.

The rest of this paper is organized as follows. Section 2 describes the CT framework, and Section 3 presents the requirements of a reliable monitor. Sections 4 and 5 study the rapidly increase of records in public logs, and the defective certificate search services of third-party monitors, respectively. Section 6 analyzes the incomplete certificate search results from the third-party monitors. Section 7 surveys the related work, and Section 8 concludes this paper.

## 2 CERTIFICATE TRANSPARENCY

The CT framework [28, 46] records CA-issued certificates in publicly-visible logs. The goal is to make it impossible for a CA to issue TLS server certificates for a domain while keeping them invisible to the claimed domain owner. As shown in Figure 1, CT introduces the following components, in addition to the traditional PKI system:

**Log server.** A log server maintains append-only logs that record certificates. The logs are publicly-visible, and anyone can fetch certificates from the logs. The records in a log are organized as a Merkle hash tree, and the root node is periodically signed by the log server, called the signed tree head (STH).

**Monitor.** Monitors regularly watch for suspicious certificates in the public logs. A monitor fetches records from the logs, decodes the certificates, and checks certificates of interest. A domain owner

[1]There are 8 third-party monitors in the Internet, but 3 of them do not provide active certificate search services now.

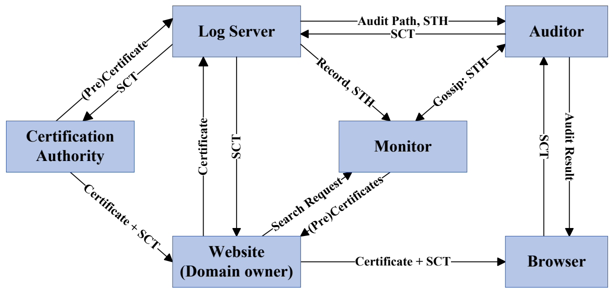


**Figure 1: The framework of certificate transparency**

may assume the monitor role to search for certificates of interest, and there are also third-party monitors which process the records in public logs to provide certificate search services for users.

**Auditor.** Auditors ensure the proper behaviors of log servers. An auditor can be a stand-alone service, or a component of TLS clients or monitors. By comparing two STHs, an auditor verifies whether a log is append-only, i.e., any particular version of the log is a superset of any previous version. It also verifies that each SCT corresponds to a record in the logs by verifying the audit path, the shortest list of additional nodes in the Merkle tree to compute the root node.

When a certificate is submitted, the log server responds with an SCT as a promise to append the certificate to the public log within the maximal merge delay (MMD). There are two methods to deliver the SCTs to browsers in TLS negotiations [46]. (*a*) *As TLS extensions.* After a CA issues the certificate for a website, the CA or the website submits it to obtain an SCT, which is then delivered as TLS extensions. (*b*) *Using certificate extensions.* Before a CA issues a certificate, it creates a *precertificate*, which binds the same data but is formatted in a way different from the final certificate. Then, the precertificate is submitted to return an SCT. Finally, the certificate is issued with the SCT embedded as a certificate extension. A log may record either a certificate or its corresponding precertificate, or both. For ease of presentation, we use the term *(pre)certificates* to denote the records in public logs and the raw search results returned from third-party monitors in the rest of this paper.

As a decentralized system, CT does not rely on fully-trusted log servers, instead, it employs a number of logs, auditors and monitors, to collectively ensure the trustworthiness of the system [17]. For example, gossip [10, 57] is implemented by exchanging information (e.g., STHs and SCTs) with other components (e.g., monitors, websites, and browsers), to detect the misbehavior of log servers, such as providing inconsistent views to different entities, failing to include submitted certificates within the MMD, etc.

## 3 ACTING AS A MONITOR: REQUIREMENTS AND CHALLENGES

The CT framework is proposed to enable the rapid discovery of fraudulent and misissued certificates. In this work, we follow the same threat model and assumptions as the ones adopted by CT [28, 46]. That is, (*a*) the CAs might be compromised or deceived by attackers, so the certificates submitted to the logs might be fraudulent or misissued, and (*b*) the correctness of log servers' behaviors is ensured by redundant auditors and monitors.

If there exists a vulnerability in the implementations of the monitor functionality designed by CT, the attackers would actively exploit this vulnerability to evade the detection of fraudulent certificates. Such fraudulent certificates, which are recorded in the public logs but actually invisible to the monitors, will be accepted by CT-enabled browsers in the MitM or impersonation attacks. This not only makes the CT framework unreliable, but also may induce more severe problems as CT-compliant certificates are supposed to be more trustworthy in the TLS/HTTPS ecosystem.

This work focuses on CT monitors, which are responsible for fetching certificates from the logs and watching for certificate of a domain of interest. As the monitors are essential to facilitate the detection of fraudulent certificates, we expect them to be *reliable.* Although reliability has not yet been formally defined or officially declared as a required property of CT monitors, several third-party monitor services claim to provide reliable certificate monitor functions for users [24, 35, 63] to return *all certificates of a domain* recorded in public logs (see Appendix A for details). In this paper, a reliable monitor is required to return the *complete* set of all certificates issued for the domain of interest.

This requirement of reliability poses two technical challenges. First, a monitor should fetch all certificates that are recorded in public logs. Since certificates are required to be duplicated among public logs [8, 17, 30] and the amount of records is increasing dramatically over time, the monitor may not fetch records from all public logs. It may alternatively select an appropriate set of logs to monitor, which provides certain guarantee to the completeness of records and the timeliness of the processing. Moreover, a third-party monitor should return the complete set of all valid (or unexpired) certificates related to any inquired domain name and also its subdomains, which are bound in the (pre)certificates in different forms (e.g., as a wildcard subdomain name), based on the records it fetches.

## 4 THE (PRE)CERTIFICATES IN PUBLIC LOGS

Many organizations, such as CA companies and Google, operate public CT logs. This section studies the public logs, especially the (pre)certificates they maintain, to understand the requirements imposed to an entity assuming the monitor role, from the perspectives of storage and network capacities.

### 4.1 Public Logs

We created a list of public logs in October 2018. It includes a total of 88 logs collected from the list maintained by Google [32], and the websites of CA companies and third-party monitors. From the type of (pre)certificates it records [32], a log server falls into one of the four categories: (*a*) *trusted-cert* (72 logs in this category), the logs run for certificates trusted by common TLS clients; (*b*) *untrusted-cert* (only one), for certificates not trusted by common TLS clients; (*c*) *expired-cert* (only one), for expired certificates; and (*d*) *testing* (14), for testing purposes only.

Meanwhile, based on the running status [20, 60], the 88 logs are classified as: (*a*) *good* (40 logs in this category), the servers are running normally; (*b*) *ceased* (19 logs), they are inactive and no longer running; (*c*) *frozen* (3), the servers are up, but do not accept new (pre)certificates any more; (*d*) *warning* (13), the logs are running but with some errors; and (*e*) *pending* (13), they are not monitored by any third-party monitor. Frozen and warning logs hold a large number of (pre)certificates and still provide services, while pending logs are either not publicly announced or for testing purposes only, which are not recommended for regular use [60].

The above categorizations distinguish the public logs according to their functional and operational status. Next, we define two sets of logs that need further investigation.

**Regular logs.** We call the logs that accept trusted certificates (i.e., trusted-cert logs) and are currently active (i.e., logs in good, frozen or warning status) as *regular logs*. To be accepted by CT-enabled browsers/platforms, a certificate must be publicly-visible in some regular log. This set denotes the *maximal* set of logs that a reliable monitor needs to monitor. Among the 88 public logs, 50 logs are included in this set, which consists of 16 Google-operated logs and 34 non-Google-operated ones.

**Google-approved log.** Chrome, the first browser that supports CT, pre-installs the public keys of 32 logs to verify the SCTs in TLS negotiations [32]. Among these 32 logs, 6 are now disqualified – five were ceased and one is still running but disqualified. They are still pre-included in Chrome mainly for backward compatibility. Therefore, we focus on the remaining 26 qualified logs in this work, and denote them as *Google-approved logs*.

## 4.2 The Rapid Increase of (Pre)Certificates

We collected the historical STHs of each log, which are archived in SSLMate [61], to explore the amount of records in public logs. Figure 2 shows the numbers of (pre)certificates in the sets of regular logs and Google-approved logs from January 1 to October 27, 2018 (42 weeks in total). Note that these numbers are extracted directly from STHs of the logs, and duplicated (pre)certificates are counted.

**Regular logs.** Figure 2(a) shows the cumulative growths of records in regular logs, Google-operated regular logs, and non-Google-operated regular logs, respectively. Until Oct. 27, 2018, 2,771,590,678 (pre)certificates are recorded in 50 regular logs, at an average growth of 6,275,652 records per day. This data set contains a large number of duplicate certificates, since the CT policies require each certificate to be submitted to multiple logs [8, 30]. As shown in Figure 2(a), the amount in regular logs increases relatively slowly from January to May 2018, at an average rate of 4,721,304 records per day. The number of records in public logs was rather small until 2016 [67]. Since June 2018, the average growth rate significantly increases to 7,778,870 records per day, with about 65% in Google-operated logs. This rapid increase since June 2018 was also reported in a recent study [70]. It is consistent with the mandatory enforcement of the CT policies on Chrome and Apple platforms in 2018 that only the CT-compliant certificates will be marked as trusted [8, 29].

We randomly sampled 42,752 records from these logs and the average size of each record is about 5.93 KB. So, storing 2,771,590,678 (pre)certificates from all 50 regular logs requires at least 15.31 TB of storage space. Moreover, with the average growth rate of 7,778,870 records per day, it requires an additional storage of 43.99 GB per day to store the newly appended (pre)certificates. At the same time, downloading only the new (pre)certificates from regular logs demands a 5Mbps network bandwidth dedicated to this task.

**Google-approved logs.** Monitoring all 50 regular logs imposes a requirement of very large storage and network bandwidth capacities to a monitor, which is probably beyond the capacity of ordinary domain owners. A reliable monitor may choose to monitor only a subset of important logs, such as the Google-approved logs.

Until October 27, 2018, there are 2,639,608,856 (pre)certificates in total in the 26 Google-approved logs, which is about 95%[2] of the records in all 50 regular logs. As shown in Figure 2(b), the average growth of certificates in Google-approved logs is about 5,928,983 records per day. In particular, since June 2018, 7,242,755 records (about 40.96 GB) on average are appended every day to these logs. Therefore, monitoring only the Google-approved logs still requires very huge storage and network bandwidth capacities.

**Google-operated & Google-approved logs.** Among the 26 Google-approved logs, 9 are operated by Google and the other 17 are not. According to the Chrome CT policy [30], a CT-qualified certificate is recorded in at least one Google-operated log and one non-Google-operated log. Thus, to be accepted by Chrome (or other TLS software adopting this policy), a TLS server certificate has to be recorded in some Google-approved log that is operated by Google.

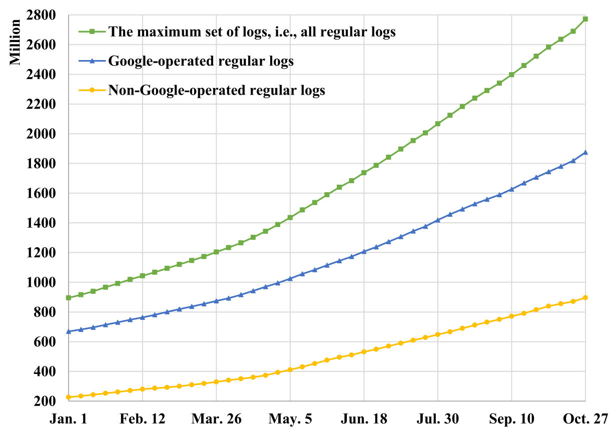
This set of 9 Google-operated & Google-approved logs denotes the *minimal* set of public logs that need to be monitored by reliable monitors, for the certificates compliant with the Chrome CT policy. Less than 0.2% websites adopting the CT framework but *not* complying with this policy [70], so this set works for almost all websites in the wild. However, as shown in Figure 2(b), monitoring this minimal set still consumes huge storage space and network bandwidth. There are 1,866,390,690 (pre)certificates in these logs by October 27, 2018. In particular, since June 2018, the amount increases at a daily growth of 5,002,599 records (i.e., about 28.29 GB per day).

The adoption of CT by browsers may result in various CT policies and also different sets of approved log servers [44]. Chrome [29], Apple [8] and Mozilla [55] publish different CT policies and maintain different sets of approved logs. This work focuses on the CT policy of Chrome, to derive the minimal set of Google-operated & Google-approved logs. On the contrary, the CT policies of both Apple and Mozilla require the set of *all* Google-approved logs to be the minimal set that needs to be monitored, because a certificate with any two or more SCTs by approved log servers is defined as CT-compliant [8, 55].

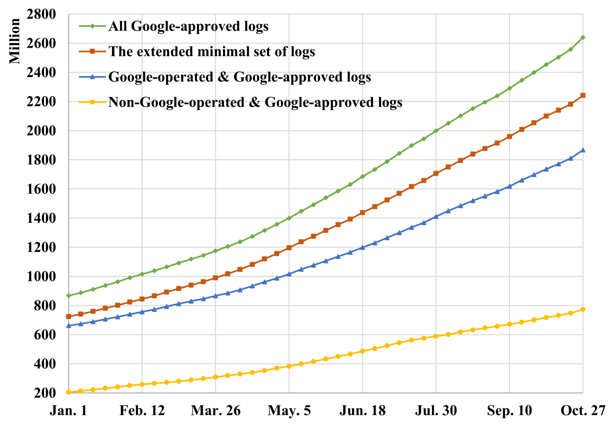
**Summary.** It requires a monitor to process on average 7,778,870 (pre)certificates at a size of 43.99 GB per day for monitoring all regular logs, or about 28.29 GB per day for the minimal set. As each certificate has its validity period, the renewed certificates will be constantly submitted to the logs. Moreover, the amount of records will keep increasing as the wide adoption of CT in the future.

The large amounts of (pre)certificates bring enormous challenges for an ordinary domain owner which assumes the monitor role, including (*a*) the huge capacities of storage and bandwidth, and (*b*) the timely processing of the rapidly-increasing data. We believe, the large requirements of storage, bandwidth and processing prevent an ordinary domain owner from acting as the monitor by itself. In fact, even some third-party monitors have problems in

[2]The data reported here are before deduplication, so the certificates that are not maintained by Google-approved logs are actually much fewer than 5%.



(a) The (pre)certificates in 50 regular logs.



(b) The (pre)certificates in 26 Google-approved logs.

**Figure 2: The rapidly-increasing records in the public logs.**

meeting the second requirement and have to keep lots of fetched-but-unprocessed (pre)certificates in backlogs; for example, some (pre)certificates have been kept in backlogs by crt.sh for a long period of time (several days or even over a year) [12].

## 4.3 The CAs Accepted by Public Logs

Mainstream browsers and platforms pre-install a number of root CAs that they trust by default, called *mainstream CAs* in this paper. There are 371 root CA certificates in Microsoft Windows, 148 in Mozilla NSS,[3] and 168 in Apple macOS. The union consists of 386 root CAs [7, 52, 54].[4] Next, we study the coverage of mainstream CAs by logs and derive a practical minimal set of logs that we recommend the monitors to monitor.

*4.3.1 The support to mainstream CAs.* Each log holds an accept list of CAs and accepts only the (pre)certificates issued by these CAs. Using the *get-roots* command [46], we obtain the list of root CAs accepted by each log. Table 1 lists the number of unique CAs accepted by different sets of logs (denoted as $\#_{CA}$) and the number of mainstream CAs *not* accepted by them (denoted as $\#^{\ominus}_{CA}$).

As shown in Table 1, 50 regular logs support 581 unique root CA certificates in total, among which 381 belong to mainstream CAs. While 200 root CA certificates accepted by the regular logs do not belong to current mainstream CAs, 5 mainstream CAs are *not* accepted by any regular log. Most of these 200 certificates are expired certificates of mainstream CAs. A log operator rarely removes expired CA certificates from the accept list [36], mainly for compatibility purposes. Meanwhile, the log operators may include some CAs trusted by some platforms other than Microsoft Windows, Mozilla NSS, and Apple macOS [36].

[3]In the certificate trust list provided by Mozilla, the *intermediate* CA "GlobalSign Extended Validation CA - SHA256 - G2" is tracked as a trust anchor, but not in the root CA list of others. Its parent root CA is trusted by Microsoft Windows and Apple macOS, which is accepted by Google-operated & Google-approved logs. We ignore this intermediate CA, because its parent root CA has been counted.

[4]Chrome uses the CA list of Mozilla NSS as its certificate trust list on Linux, and on Windows it directly adopts the root CA storage of operating system.

**Table 1: The CAs accepted by log servers.**

| Log server set | $\#_{Log}$ | $\#_{CA}$ | $\#^{\ominus}_{CA}$ |
|---|---|---|---|
| Regular | 50 | 581 | 5 |
| Google-approved | 26 | 580 | 5 |
| Google-operated | 25 | 727 | 9 |
| Google-operated & Google-approved | 9 | 537 | 9 |
| The extended minimal set | 9+2 | 542 | 5 |

However, it is problematic if mainstream CAs are not accepted by the regular logs or the Google-operated regular logs. Totally 9 mainstream CAs are not accepted by any Google-operated log (see Table 10 in Appendix C for details), and 5 out of these 9 CAs are not accepted by any of the regular logs. However, because the Chrome CT policy requires that, in TLS negotiations, at least one SCT is signed by Google-operated logs [30], the certificates issued by these 9 mainstream CAs will always be rejected by Chrome and any other CT-enabled software adopting this policy. On the other hand, the certificates issued by these 5 mainstream CAs which are unsupported by any regular log, will be rejected by Chrome always, but acceptable to browsers that currently do not support CT such as Microsoft IE and Edge. These mainstream CAs are excluded from the CT framework, so any fraudulent certificates issued by them will not be detected by monitors. It is unclear why they are not accepted by the regular logs, but it raises potential issues that need to be addressed to facilitate the wide adoption of CT.

**Summary.** The CT framework improves the accountability of CAs, but its coverage of CAs is still insufficient. Although it is widely agreed that a log servers operating in the public interest should accept any publicly-trusted CA certificate pre-installed in major browsers [36, 46], some publicly-trusted mainstream CAs are not accepted by the public logs. To improve the accountability of the TLS/HTTPS ecosystem, the 9 mainstream CAs that are currently not accepted by Google-operated logs shall be accepted by some of them in the future. Meanwhile, this will inevitably impose higher requirements on reliable monitors, because more (pre)certificates will be recorded in public logs as more CAs are accepted.

*4.3.2 The extended minimal set of logs.* As discussed above, the minimal set of logs should be extended to support more mainstream CAs that are currently not accepted. In particular, among the 9 unsupported CAs, 4 of them are accepted by non-Google-operated regular logs but not by Google-operated ones. To support these publicly-trusted mainstream CAs, we include some non-Google-operated regular logs into the minimal set.

A CA accepted by a log does not mean all (pre)certificates issued by the CA will be recorded in the log. It only means, if a (pre)certificate issued by the CA is submitted to the log, it will be accepted. Thus, in order to fetch as complete (pre)certificates issued by certain CAs as possible, we need to monitor more active regular logs. Since (pre)certificates are duplicated across the non-Google-operated logs, we consider two criteria for selecting logs into the set: (*a*) the logs with larger daily growth of appended records; and (*b*) the logs supporting the mainstream CAs that are not currently accepted by the 9 logs in the minimal set.

We finally identify two non-Google-operated logs to be included into the *extended minimal* set, as shown in Table 1. If either one of these two logs is removed from this set, some mainstream CAs will not be accepted. Meanwhile, adding more non-Google-operated logs into this set will not increase the total number of accepted mainstream CAs. Therefore, the final extended minimal set consists of 9 Google-approved & Google-operated logs and 2 non-Google-operated logs, which are also Google-approved. By October 27, 2018, this extended minimal set of logs maintains 2,241,161,280 records in total, and since June 2018, on average 6,096,426 (pre)certificates (about 34.48 GB) per day are submitted to these 9+2 logs.

## 5 THE INCOMPLETE CERTIFICATES RETURNED BY THIRD-PARTY MONITORS

In this section, we investigate the third-party monitors and evaluate the completeness of their certificate search services.

### 5.1 Third-Party Monitors

Our study on public logs indicates that it is costly and impractical for ordinary domain owners to act as CT monitors by themselves. A more practical solution is to rely on professional third-party monitors to maintain a *complete* copy of all certificates recorded in public logs and return a *complete* set of all certificates related to a domain name inquired by the domain owner.

A third-party monitor usually provides the running status of public logs, including STH, uptime, certificate scope, etc. [46], and some of them provide certificate search services. They fetches (pre)certificates from public logs, and allow the users (e.g., a domain owner) to search for certificates of interest.

To our best knowledge, there are 9 third-party monitors in the Internet, namely crt.sh, SSLMate, Censys, Google Monitor, Facebook Monitor, CT-Observatory, Edgecombe, Merkle Town and Hardenize. Among them, CT-Observatory was suspended since September 2018, and Hardenize is still under construction until January 2019. Among the remaining 7 monitors, Edgecombe and Merkle Town provide only the running status of logs but not the search service. So, we focus on the remaining 5 monitors in this work.

**Table 2: The incomplete results from Facebook Monitor.**

|  | $\#_{pre}$ | $\#_{final}$ | $\#_{unique}$ | $\#_{domain}$ |
|---|---|---|---|---|
| Certificate set 1 | 22 | 25 | 33 | 8 |
| Certificate set 2 | 13 | 45 | 48 | 14 |

$\#_{pre}$ and $\#_{final}$ denote the number of *precertificates* and *final certificates* returned in the raw results, out of the 51 (pre)certificates in logs;
$\#_{unique}$ denotes the number of unique certificates in the deduplicated results;
$\#_{domain}$ denotes the number of domains with incomplete results, out of the 17 domains.

### 5.2 Exploring the Certificate Search Services Tentatively

We conducted a small-scale experiment, by our domains with known certificates as the controlled input, to explore the search services of 5 third-party monitors. First, we applied two sets of certificates for 17 subdomains of two second-level domains (i.e., warnings.xyz and wclcttest.cn) from Let's Encrypt [47]. There are in total 102 certificates, and 3 certificates for each subdomain.[5] We searched for these certificates in 5 monitors to find whether and when the monitors return a copy of them.

All 102 certificates (in particular, 102 final certificates and 102 precertificates) are returned from crt.sh, SSLMate, Censys, and Google Monitor. Every (pre)certificate is returned within less than 8 hours after it was submitted to the logs. However, Facebook Monitor does not return all certificates until February 6, 2019 (about four weeks after the certificate issuance). As shown in Table 2, 18 certificates are missing in the first set, and 3 are missing in the second.[6] Also, the processing delay in Facebook Monitor is very large – some (pre)certificates are returned after it has been submitted to the logs for about 55 hours.

This tentative small-scale experiment, exposes a critical problem of existing third-party monitors (e.g., Facebook Monitor) that they may not return all the certificates issued for a certain domain. This motivates us to conduct a systematic study on the certificate search services and uncover potential issues if any.

### 5.3 Searching for the Certificates of Alexa Top Websites

To systematically assess the performance of the search services of 5 monitors, we conducted two sets of experiments to search for the certificates issued for Alexa Top-1K and Top-1M websites, respectively. The rationale is to study if each monitor returns a complete set of certificates for popular domains (with hundreds of certificates per domain) and ordinary domains (with 1–100 certificates per domain).

*5.3.1 The certificate search services.* Next, we describe the certificate search interfaces of the 5 third-party monitors under study.
**crt.sh.** The SQL interface is provided. The search is performed by comparing (*a*) commonName (CN) and organizationUnitName (OU)

[5] Let's Encrypt enforces an upper limit of the certificates that a domain owner can apply in one week. So, we applied the first set of 51 certificates on January 8, 2019, and the second set on January 14, 2019.
[6] We contacted Facebook for this problem on January 17, 2019, but did not receive any meaningful reply. We regularly checked the results returned by Facebook Monitor and found it returned all 102 certificates after May 2019. This indicates a processing delay of more than four weeks.

in the subject field, and (*b*) dNSName, emailAddress and iPAddress in the certificate extension of subject alternative name (SAN).

**SSLMate.** It provides an HTTP GET/POST API, and the certificate search is performed by comparing CN in the subject field and dNSName in the SAN extension.

**Censys.** The Python API is provided. Similar to crt.sh, Censys allows flexible combinations of the subject field and the SAN extension, on which the comparisons are performed.

**Google.** It does not provide any API, and we have to access the service on web pages. We developed a tool to access the URLs, and parse the results on web pages. The search is performed by comparing CN and dNSName.

**Facebook.** The Facebook Graph API is provided for developers to access Facebook Monitor. The search is performed by comparing CN and dNSName.

While providing free services, the monitors enforce limitations. For example, Censys supports only 250 searches per month, 0.4 action per second, and up to 1,000 records per search for free. Meanwhile, it offers commercial plans. We adopt Censys Pro Plan (25,000 searches per month, up to 25,000 records per search, and 1.0 action per second) in the experiments. Facebook Monitor dynamically controls the search speed based on the number of concurrent users.

#### 5.3.2 *Dealing with different search policies.*

When a domain name is input, we expect a third-party monitor to return all unexpired certificates binding the inquired domain name and its subdomains, in any valid form (e.g., as a wildcard subdomain name).

However, five monitors return very different results for a same input of domain name, due to the very different search policies they adopt. For fair comparisons and systematical investigations, we need to explore the differences in their search policies, and design strategies for each monitor to conduct customized searches for each inquiry. This ensures the results include all the related (pre)certificates from the monitors.

**Certificates expected to return.** We specify the related certificates of a domain that are expected to be returned. This specification helps us to fairly compare the results returned from five monitors and systematically assess their completeness. Given a domain name such as "B.A", we expect the result includes all certificates in their validity periods and binding any of the following types of domain names in CN or dNSName, in a case-insensitive way: (*a*) the domain name inquired (i.e., B.A); and (*b*) any subdomain name with wildcard or not (i.e., *.B.A, X.B.A, Y.B.A, Z.B.A, WWW.B.A, etc.)

In addition, if the inquired domain is a third-level or higher-level domain (e.g., C.B.A) and its parent domain (i.e., B.A) is not in the top-level domain (TLD) list [78], (*c*) any certificate binding the related wildcard domain name (i.e., *.B.A) is also expected to be returned. For example, Alexa Top-1K websites consist of 893 second-level domains and 107 third-level ones, among which 8 third-level domains have a parent domain not in the TLD list. For these domains, we expect the certificates with the corresponding wildcard domain names (e.g., "*.tumblr.com" for "media.tumblr.com").

**The search policies of third-party monitors.** The monitors do not disclose their search policies in details. We had to exhaustively explore a variation of combinations of comparison statements to infer the policies, e.g., with upper/lower-case (wildcard) domain names of different levels, available options enabled/disabled, etc. We show our findings in Table 3 to provide an overview of the search policies adopted by the five third-party monitors. In particular, *When a domain name (e.g., B.A) is input without an explicit subdomain option,* in addition to the certificates binding that domain name, Censys and Facebook Monitor also return the ones binding subdomain names (i.e., X.B.A, *.B.A, etc.). If it is a third-level or higher-level domain (e.g., C.B.A), Google Monitor also returns the certificates for the related wildcard subdomain name (i.e., *.B.A). *When a domain name is input with the subdomain option (if supported),* crt.sh and Google Monitor return only the certificates binding the subdomain names (i.e., X.B.A, *.B.A, etc.), but not the input domain name, while SSLMate returns the ones binding the domain name (i.e., B.A) and any subdomain name (i.e., X.B.A, *.B.A, etc.). If it is a third-level or higher-level domain, SSLMate also returns the certificates binding the related wildcard domain name (i.e., *.B.A). *When a wildcard domain name is input for the certificates of a domain (e.g., we input *.B.A to search for the certificates of C.B.A),* Facebook Monitor returns the certificates binding any domain name matching the wildcard input (i.e., X.B.A, Y.B.A and even W.Z.B.A), while others return only the ones exactly binding the wildcard domain name (i.e., *.B.A).

**Customized comparison statements for five monitors.** We composed the comparison statements customized for each monitor according to its search policy. As shown in Table 3, to return the expected parts of related certificates for each inquired domain name, we need to conduct (*a*) one search in SSLMate, Censys and Facebook Monitor; (*b*) two searches in crt.sh and Google Monitor; and (*c*) one additional search in crt.sh, Censys and Facebook Monitor for a third-or-higher-level domain whose parent domain is not in the TLD list. Besides, we need to filter out the *unexpected* part by ourselves, after the addition search in Facebook Monitor.

**Other pre- and post-processing.** The search services of all monitors except Censys are not case-sensitive. Censys is case-insensitive if the inquired domain name is lower-case, but if the input contains any upper-case character, it becomes case-sensitive. So, we converted all inputs into lower-case in our experiments.

crt.sh, SSLMate, Censys, and Google Monitor implement the option to exclude expired certificates, while Facebook Monitor does not. But Censys works with delays in marking an expired certificate, which causes some newly-expired (for one or two days) certificates to appear in the results. So, we need to filter out the expired certificates returned from Censys and Facebook Monitor.

Censys returns certificates with a CN containing the string of the input domain name, even when CN is not a domain name. For example, when we search the domain sohu.com, the certificate with CN "Developer ID Application: Sohu.com Inc. (NASDAQ: SOHU) (X3XWZ5HCGK)" appears in the result. In the experiment of Alexa Top-1K websites, 7 certificates are returned for 5 domain names due to this feature. In addition, the results from Censys include some non-publicly-trusted or testing certificates, which are recorded in the untrusted-cert or testing logs it monitors. Such certificates are not accepted by browsers or returned by other monitors, so we filtered them out before the analysis.

#### 5.3.3 *The incompleteness of search results for Alexa Top-1K websites.*

We conducted the experiment to search for certificates issued for Alexa Top-1K websites on October 27, 2018. After filtering out

**Table 3: The service interfaces and search policies of the third-party monitors.**

| | Input vs. Result† | | | | Comparison | Valid/Expired | Case- |
|---|---|---|---|---|---|---|---|
| | B.A | B.A w/ subdomain opt. | *.B.A | API | Scope | Option | Insensitive |
| crt.sh | –a– | –b– | —c– | ✓ | Customized | ✓ | ✓ |
| SSLMate | a— | abc | —c– | ✓ | CN & dNSName | ✓ | ✓ |
| Censys | ab– | – | —c– | ✓ | Customized | ●1 | ●2 |
| Google Monitor | a–c | –b– | —c– | – | CN & dNSName | ✓ | ✓ |
| Facebook Monitor | ab– | – | abcd | ✓ | CN & dNSName | – | ✓ |

✓: The feature is supported. ●: This feature is partially supported. –: The feature is unsupported, or this part of related certificates is not returned.
†: For a domain, a monitor returns different combinations of some related certificates binding (*a*) the domain name, which is expected; (*b*) any subdomain name, either wildcard or not, expected; (*c*) the related wildcard domain name, expected if the searched domain is third-level or higher-level and its parent domain is not in the TLD list; and (*d*) any unrelated domain or subdomain name matching the input wildcard domain name (e.g., we input *.B.A for C.B.A, but it returns X.B.A, Y.B.A and even W.Z.B.A, etc.), which is unexpected.
1: Censys filters out expired certificates in the results, but it works with delays in marking newly-expired certificates.
2: The service of Censys is case-insensitive, if the input is lower-case; if the input contains any upper-case character, it becomes case-sensitive.

**Table 4: The results for Alexa Top-1K websites.**

| | $\#_{cert}$ | $\#_{unique}$ | $P_n$ | $\#_{domain}$ |
|---|---|---|---|---|
| crt.sh | 407,660 | 327,019 | 14.4% | 104 |
| SSLMate | 201,954 | 201,954 | 47.1% | 164 |
| Censys | 418,382 | 333,993 | 12.6% | 120 |
| Google Monitor | 268,152 | 181,664 | 52.3% | 546 |
| Facebook Monitor | 327,805 | 252,189 | 34.0% | 289 |

$\#_{cert}$: the number of valid (pre)certificates searched;
$\#_{unique}$: the number of unique certificates, after we deduplicate the results;
$P_n$: the proportion of unique certificates *not* returned, compared with the reference sets;
$\#_{domain}$: the number of domains with *incomplete* results, compared with the reference sets (excluding the 18 domains with no certificate).

**Table 5: The number of domains with incomplete results in each group of websites.**

| $\Phi$ | = 1 | (1, 10] | (10, 100] | > 100 |
|---|---|---|---|---|
| $D_\Phi$, $\sum$ = 982 | 50 | 235 | 471 | 226 |
| crt.sh | 0 | 8 | 28 | 68 |
| SSLMate | 0 | 7 | 53 | 104 |
| Censys | 0 | 3 | 33 | 84 |
| Google Monitor | 0 | 59 | 269 | 218 |
| Facebook Monitor | 0 | 14 | 102 | 173 |

$\Phi$: the range of the number of certificates for a domain.
$D_\Phi$: the number of domains in a group of websites. For every domain in this group, the sum of unique certificates returned is within the range $\Phi$.

invalid TLS server certificates such as expired certificates and code-signing certificates, we obtain a raw dataset with a total of 1,623,953 valid (pre)certificates from 5 third-party monitors.

**Deduplicated unique certificates.** The raw search results from crt.sh, Censys, Google and Facebook Monitors may include a certificate and its corresponding precertificate at the same time, while SSLMate returns deduplicated results (either the certificate or the equivalent precertificate, but not both). Although a precertificate is invalid in TLS negotiations, it corresponds to a server certificate and the misissuance of precertificates is considered equivalent to the misissuance of the final certificate [46]. So we treat a certificate and its corresponding precertificate as *equivalent* to each other. We define the four-tuple (*NotBefore, NotAfter, SerialNumber, Issuer*) as the index to identify a (pre)certificate, and deduplicate the data from each monitor as well as the union of all searched certificates. Finally, we obtain a dataset of 382,051 deduplicated unique certificates in total for Alexa Top-1K websites.

**The reference set.** Before analyzing the completeness of the returned results, we face a challenge that it is difficult to obtain the ground truth data about the complete set of valid certificates issued for each inquired domain. In fact, if such ground truth for any domain is easily available, the CT framework is not needed to detect fraudulent certificates. So, we define a reference set to approximate the complete set, which is the union of the certificates returned by all five monitors for each inquired domain. Note that conceptually this reference set is only a subset of the real complete set.

**Results for Alexa Top-1K websites.** We compare the certificates returned from each monitor against the reference set to assess its completeness. For 18 domains among the 1,000 domains, no (pre)certificate is returned by any monitor. We accessed these 18 websites manually, and confirmed that they did not hold valid certificates in public logs (see Appendix B for details). For the other 982 domains, no monitor returns the complete set of known certificates. The results are shown in Table 4.

We assess the performance of five monitors from two aspects, (*a*) the proportion of unique certificates *not* returned; and (*b*) the number of domains of which the searched certificates are *incomplete*. Google Monitor is the worst, which fails to return 52.3% of the certificates for 546 domains. Censys yields the best result in terms of the numbers of certificates returned (87.4%), while crt.sh is the best in terms of the numbers of domains with complete results (878 domains). However, even the best services miss 12.6% of certificates and return defective results for 104 out of 982 domains.

Finally, we explore the relationship between the number of certificates a domain holds and the incompleteness of the results. We divide the 982 websites into 4 groups according to the number of valid certificates each domain holds in the reference sets. As shown in Table 5, there are 50 domains with only one certificate, 235 domains with more than 1 but less than 10 certificates, 471 domains with more than 10 but less than 100 certificates, and 226 domains with more than 100 certificates. For each group, Table 5 shows the number of domains with incomplete results from each monitor.

**Table 6: The results for Alexa Top-1M websites.**

| | $\#_{cert}$ | $\#_{unique}$ | $\#_{domain}$ | $\#_{average}$ |
|---|---|---|---|---|
| Top-1K | 1,623,953 | 382,051 | 982 | 389.05 |
| Top-(1K, 5K] | 770,257 | 114,061 | 968 | 117.83 |
| Top-(5K, 20K] | 341,202 | 48,869 | 938 | 52.10 |
| Top-(20K, 100K] | 146,863 | 19,558 | 878 | 22.28 |
| Top-(100K, 500K] | 72,610 | 9,669 | 834 | 11.59 |
| Top-(500K, 1M] | 45,546 | 6,462 | 772 | 8.37 |

$\#_{cert}$: the number of valid (pre)certificates searched;
$\#_{unique}$: the number of unique certificates;
$\#_{domain}$: the number of domains which return at least one (pre)certificate;
$\#_{average}$: the average number of certificates per domain, i.e., $\#_{unique}$ / $\#_{domain}$.

In general, more popular domains (with more certificates) have larger probabilities of incomplete results. Nearly half of the domains have 10–100 certificates. The probability that a domain in this group receives an incomplete result is 26.9%, 32.3%, 27.5%, 49.3%, and 35.3%, from five monitors, respectively. On the other hand, for each of the 50 domains with only one certificate, its certificate is searched successfully by all five monitors. However, for other three groups, no monitor returns the complete results for all domains. Note that, if fraudulent certificates are issued for a domain, the number of certificates related to the domain name will be greater than one.

*5.3.4 The results for more ordinary domains.* Based on the above analysis, we extended our study to more ordinary domains. We conducted the second experiment on January 6, 2019, and randomly selected 1,000 domains from each of the five segments of Alexa Top-1M websites: Top-(1K, 5K], Top-(5K, 20K], Top-(20K, 100K], Top-(100K, 500K], and Top-(500K, 1M].

After the same pre- and post-processing, we collected a total of 1,376,478 (pre)certificates for the 5,000 websites. We deduplicated the data and obtained the reference sets with 198,619 unique certificates. The statistics are shown in Table 6. These are less popular domains, with fewer certificates per domain. In particular, in the Top-(500K, 1M] segment, only 772 out of 1,000 websites hold certificates, and on average each holds less than 9 certificates.

We compare the results returned from each monitor with the reference sets. Table 7 lists the proportion of unique certificates that are *not* returned from each monitor. In general, the probability of missing certificates decreases for less popular websites. Censys and crt.sh perform better than others, and in the best case only 0.1% of certificates are not returned by Censys in the searches of Alexa Top-(100K, 500K] websites. Google Monitor is the worst, and at least 6.7% of certificates are missing in these experiments.

Table 8 summarizes the number of domains with incomplete results for each segment. Censys performs better than others, but there are always incomplete results for some websites in each segment of Alexa Top-1M websites. In the best case, 7 out of 1,000 domain names return incomplete certificates from Censys. Facebook Monitor is the worst, and out of the 1,000 domains in each segment, there are always hundreds of domain names not returning complete certificates. We further study the domains with only one certificate and the ones with less than 10 certificates to check if the monitors have any tendency towards missing the certificates of less popular domains (see Table 11 in Appendix C for details). For less popular sites, Facebook Monitor misses more certificates than others. For ordinary domains, all monitors perform similarly. Overall, Facebook Monitor always returns incomplete results for about 15% of domains, while Censys performs the best and returns incomplete results for only a few domains of each segment.

### 5.4 Summary and Discussion

Our experiment results uncover that *none* of the five active third-party monitors provides reliable certificate search services that guarantee to return the *complete* set of certificates for the inquired domain name. All five monitors demonstrate defects in their certificate search services, in the experiments with both popular websites and less popular websites. This exposes a critical problem that will degrade the reliability of not only the monitor but also the CT framework. If a fraudulent certificate issued for a certain domain is not returned from the certificate search, it would never be detected.

This problem may be more severe than we expose as above, because the reference sets we used are only the approximations of the real complete sets for there is no available ground truth data for Alexa Top websites. That is, *even the union of results from 5 third-party monitors could not guarantee to include complete certificates for a certain domain.* To illustrate this problem, we construct the base reference sets for Alexa Top-1K websites with only the results from crt.sh, and add the results from SSLMate, Censys, Google Monitor and Facebook Monitor one by one to enlarge the reference sets. The number of deduplicated unique certificates increases from 329,019 to 334,605, 373,634, 376,308 and finally 382,051. Correspondingly, the number of domains with complete results increases from 878 to 888, 949, 954 and finally 982. We argue that, when there are more active third-party monitors (such as CT-Observatory and Hardenize if they provide services in the future), or even when the monitors monitor more logs and process more (pre)certificates, the reference sets will probably be larger than the ones in our experiments.

## 6 THE CAUSES OF INCOMPLETE CERTIFICATE SEARCH RESULTS

We discuss some causes of the incomplete results from five monitors and propose some suggestions for more reliable monitors.

### 6.1 Potential Causes due to Unmonitored Logs

Since each (pre)certificates is required to be submitted to multiple logs [8, 30] and the public logs maintain a great number of duplicated (pre)certificates, it may not be necessary for a third-party monitor to monitor all logs for efficiency and cost reasons. So, we investigate the set of logs that each monitor (un)monitors to check if the log coverage causes incomplete results. While we believe the third-party monitors, especially the ones with certificate search services, should provide the list of logs they monitor to the public to help assess the quality of their services [44], Google and Facebook Monitors do not disclose such information. In Table 9, we summarize the numbers of logs in five different sets that are *not* monitored by each of the remaining 6 monitors.

First, no monitor covers all 50 regular logs, and only SSLMate and Edgecombe monitor all 26 Google-approved logs. In Sections 4.2 and 4.3, we define the extended minimal set of logs as a reference set for third-party monitors to balance the requirement of reliability

**Table 7: The proportion of unique certificates *not* returned, for each segment of Alexa Top-1M websites.**

| | Top-1K | Top-(1K, 5K] | Top-(5K, 20K] | Top-(20K, 100K] | Top-(100K, 500K] | Top-(500K, 1M] |
|---|---|---|---|---|---|---|
| crt.sh | 14.4% | 3.9% | 0.3% | 0.4% | 0.4% | 0.8% |
| SSLMate | 47.1% | 10.5% | 1.3% | 0.7% | 0.4% | 0.9% |
| Censys | 12.6% | 0.3% | 1.0% | 0.5% | 0.1% | 0.2% |
| Google Monitor | 52.3% | 22.2% | 16.2% | 10.2% | 8.6% | 6.7% |
| Facebook Monitor | 34.0% | 12.9% | 5.0% | 5.3% | 5.7% | 6.0% |

**Table 8: The number of domains of which the results are *incomplete* compared with the reference sets, among the 1,000 domains randomly-selected from each segment of Alexa Top-1M websites.**

| | Top-1K | Top-(1K, 5K] | Top-(5K, 20K] | Top-(20K, 100K] | Top-(100K, 500K] | Top-(500K, 1M] |
|---|---|---|---|---|---|---|
| crt.sh | 104 | 78 | 46 | 29 | 16 | 11 |
| SSLMate | 164 | 100 | 61 | 33 | 19 | 15 |
| Censys | 120 | 52 | 27 | 14 | 7 | 7 |
| Google Monitor | 546 | 421 | 294 | 198 | 117 | 73 |
| Facebook Monitor | 289 | 307 | 393 | 259 | 226 | 160 |

**Table 9: The logs (un)monitored by monitors.**

| | $\#_{Log}$ | $\#_{r}^{\ominus}$ | $\#_{go}^{\ominus}$ | $\#_{ga}^{\ominus}$ | $\#_{go+ga}^{\ominus}$ | $\#_{e}^{\ominus}$ |
|---|---|---|---|---|---|---|
| crt.sh | 46 | 9 | 8 | 4 | 1 | 1 |
| SSLMate | 77 | 1 | 1 | 0 | 0 | 0 |
| Censys | 46 | 19 | 12 | 5 | 0 | 0 |
| CT-Observatory † | 20 | 39 | 20 | 18 | 4 | 6 |
| Edgecombe | 77 | 2 | 7 | 0 | 0 | 0 |
| Merkle Town | 39 | 11 | 10 | 4 | 0 | 0 |

$\#_{r}^{\ominus}$, $\#_{go}^{\ominus}$, $\#_{ga}^{\ominus}$, $\#_{go+ga}^{\ominus}$, $\#_{e}^{\ominus}$: the numbers of regular logs, Google-operated logs, Google-approved logs, Google-operated & Google-approved logs, and logs in the extended minimal set, which are *not* monitored.
†: The data were collected in August 2018 before it was shut down.

and the cost. In Table 9, we find that most monitors actually cover more logs than these sets. In particular, among the 3 monitors with active certificate search services, SSLMate and Censys monitor all 9 Google-operated & Google-approved logs as well as all logs in the extended minimal set. crt.sh., on the contrary, only monitors 8 out of 9 Google-operated & Google-approved log.

We comprehensively compare the list of logs (un)monitored by five monitors and the certificate search results in the experiments for Alexa Top websites. The (pre)certificates returned from some monitors are accompanied with SCTs. An SCT describes by which log it is signed (i.e., from which log it is fetched). We then study the search results one by one, but do not find any direct evidence that relates any missing certificate to the logs unmonitored by a monitor. For example, SSLMate has a better log coverage than crt.sh and Censys, as shown in Table 9, but it misses more certificates than the other two in our experiments, as shown in Tables 7 and 8. This may be due to the duplicated (pre)certificates across logs, which mitigates the impact of not monitoring one log.

We study the observations in the experiments with Google and Facebook Monitors, although they do not disclose the list of logs they are monitoring. In the preliminary experiments on Alexa Top-1K websites in September 2018, we found that Google Monitor did not return any record in the Argon2019 log. However, in the experiment on October 27, 2018 (in Section 5.3), some records in Argon2019 were returned by Google Monitor. It is not possible for us to tell if this is caused by any adjustment of logs monitored by Google Monitor or other issues. The log coverage cannot explain the failure of Facebook Monitor in our experiment with Let's Encrypt in Section 5.2. We find that Let's Encrypt submitted 102 precertificates to 5 logs (i.e., Argon2019 and Icarus operated by Google, and non-Google-operated Nimbus2019, Sabre and Yeti2019). Each precertificate is submitted to at least two and at most five logs, and all the 102 final certificates are submitted to Argon2019. From other (pre)certificates returned from Facebook Monitor, we can tell it is monitoring all the 5 logs. But, we cannot tell why it missed 21 certificates for more than 4 weeks in this experiment.

We also checked the behavior of log servers. Edgecombe exchanges the STHs of most public logs with Chromium STHSets [33], Google [34], and SSLMate, by the gossip validation [57]. From the public results disclosed by Edgecombe, there is no log misbehavior that provides inconsistent views to different monitors.

## 6.2 Causes due to Issues in Monitor Implementation

We analyze the results of each monitor, and discover several issues that may cause the missing records. However, since the internal mechanism and architecture of CT monitors are unknown to the public, we can only study the problem from the perspective of external users. We believe our findings only reveal a subset of causes, and more unknown bugs or vulnerabilities in the monitors' services exist. For example, all the causes cannot fully explain the missing (pre)certificates in our experiment in Section 5.2.

We have reported the issues uncovered in the paper to all five CT monitors with our experiment results and analysis. Censys replied with an explanation about potential causes due to data migration (see the remainder for details). Both crt.sh and SSLMate replied that they would initiate investigation on the reported issues, but we have not received further feedback from them. Finally, until the submission of this manuscript, we have not received any meaningful reply from Google and Facebook Monitors.

**Delayed processing**. The monitors may not be able to process fetched records in time and thus store them in so-called backlogs. We find the records from some large logs (e.g., Pliot, Rocketeer and Argon2019) are not processed in time by crt.sh. The SCTs indicate that many of the (pre)certificates have been stored in backlogs for several days and even over a year without being processed. The processing delay causes some incomplete results of crt.sh. We compare the backlogs on August 27 and October 27, 2018, and the former is much larger than the latter. This is consistent with our observation of much more missing certificates in the preliminary experiment in August 2018 – for 500 domains in the Alexa Top-1K list, the returned results were not complete. There are also delays in the service of Censys – some (pre)certificates newly-appended in the logs for one or two days are not returned.

**Unreturned precertificate**. As shown in Table 7, Google Monitor misses a nonnegligible proportion of certificates, regardless of the popularity of the domains. We find that, if a certificate is recorded in public logs only in the format of precertificate, it is probably not returned by Google Monitor. But Google Monitor does return some precertificates. For example, for Alexa Top-1K websites, Google Monitor returns incomplete results for 546 domains, among which 396 domains miss only precertificates. These missing precertificates accounts for 76.9% of all missing results of Alexa Top-1K websites from Google Monitor. This indicates the processing of precertificates in Google Monitor may be faulty.

**Incident not recovered in time.** We observe a large number of missing results are related to specific time windows. For example, the SCTs of most (pre)certificates missed by Censys have timestamps between UTC 2017-10-11/12/13/14, and Google Monitor misses many (pre)certificates whose SCTs were signed on UTC 2015-11-20. It is probably due to interruptions during data retrieval or processing. While the causes of the interruption vary, the results indicate that the services of these monitors are not fault-tolerant.

We disclosed our findings about the missing certificates to Censys. After their investigation, we were told that this might be caused by incidents during their data migration between July and August of 2018. Our later experiments showed that Censys was fixing this problem gradually – our experiment on Aug 21, 2018 discovered missing certificates for 200 domains among Alexa Top-1K websites, while on October 27, this number was reduced to 120 domains. But the recovery process is taking a longer period than we expect. Until January 2019, we still observed missing certificates with timestamps in these time windows. Besides, a small number of certificates out of these time windows were still not returned, as shown in Tables 7 and 8, for which Censys did not provide a clear explanation.

**Unsupported domain**. SSLMate misses many certificates due to its restricted services to four domains (i.e., amazonaws.com, cloudfront.net, blogspot.com and fbsbx.com). The requests about these domains are directly terminated with the error "not_allowed_by _plan". However, the certificates of these 4 domains are recorded in at least 37 logs, among which SSLMate monitors 27 at a regular basis. We find that SSLMate claims it only accepts domain names of registered domains or their subdomains [62], and thus deliberately excludes these four domains to protect user privacy (e.g., customers' hostnames in amazonaws.com, cloudfront.net, blogspot.com and fbsbx.com) [70]. Similarly, Google Monitor returns no result for blogspot.com and cloudfront.net, but it does not provide any specific explanation or return any error message.

Facebook Monitor returns the warning, "provided domain is invalid," for some domains (e.g., btrc.gov.bd and sabay.com.kh). Among 6,000 selected Alexa Top-1M websites, the requests of 9 domains are terminated with such errors. We doubt the error is *not* caused by the privacy concerns, since Facebook Monitor returns non-empty results for domains with obvious privacy problems (e.g., cloudfront.net and blogspot.com). Moreover, Facebook Monitor returns no results for many other domain names.

**Interface limitation**. Some monitors limit the number of records returned in each search. For example, Censys has a restriction of at most 25,000 records per search and Facebook Monitor sets this value to 5,000. Similarly, in SSLMate, we observe a query of zendesk.com, for which many (pre)certificates are supposed to be returned, receives the error "This query took too long to complete."

Any failure to meet this requirement causes missing results. However, several domains have certificates larger than this limit. For example, among Alexa Top-1K websites, there are 4 domains exceeding the 250,000-record limit, i.e., amazonaws.com (43,306), zendesk.com (34,341), cisco.com (39,045), and att.com (32,496), and 29 domains exceeding the 5,000 limit. When the relevant records exceed the limits, one has to issue multiple queries and combine the results, which may result in some (pre)certificates overlapped or missing.

In summary, in the experiments, the interface limitation causes a number of certificates not returned, but its impact is limited - it affects only a very small amount of domains.

## 6.3 Potential Countermeasures

Our experiments show that the monitors are not fault-tolerant, due to the scale of the (pre)certificates and the complexity in processing them. The monitors, like other CT components, should not be assumed by default as fully trustworthy, since CT is by nature a decentralized system. From these considerations, we propose to design countermeasures to improve the reliability of CT monitors.

**Reliability audit of monitor services.** CT auditors and the gossip protocol [57] are designed to detect the misbehavior of CT logs. We believe a similar audit mechanism should be implemented to detect the misbehavior or problematic behavior of CT monitors, especially on service reliability. In particular, we propose to regularly evaluate the state of online monitor services and identify the problems in these services through two black-box testing approaches.

The first black-box testing approach, similar to the experiments in this paper, is to periodically collect certificate search results for the same domain name from multiple monitors to generate reference sets for tested domains. By comparing the results from a single monitor with the results in reference sets, the auditor and the monitor can detect if any valid record is not correctly returned. The number of missing records and the number of affected domains can be used to evaluate the reliability of the services of CT monitors. The second black-box testing approach involves a set of CAs to regularly issue multiple types of certificates for a selected set of test domain names and submit them to the logs. The auditor or the monitor itself can issue queries about the test domain names to check the monitoring states of these certificates, e.g., whether they are timely monitored and whether the monitoring results are complete. This is similar to the black-box test method for checking the correctness of data query in database systems [65, 68].

**MaaS for elastic resource allocation.** With the rapidly increase of certificates in logs, a monitor has to deal with the scalability challenge and allocate its limited resources among tasks. Incapable of addressing this challenge leads to several issues as we observed in this study, such as erroneous certificate processing, delayed incident recovery, etc., which might lead to delayed or failed detection of fraudulent certificates. Therefore, we propose to enable elastic resource allocation in third-party monitors so that they can always have sufficient resources to process the great number of (pre)certificates retrieved from logs correctly.

To implement elastic resource allocation for monitors, we can possible deploy the monitor functions in a cloud computing platform, e.g., to enable "monitoring as a service (MaaS)" – certificate search and monitoring services [16]. MaaS in the cloud brings several advantages: (i) resources are allocated on demand so that certificates from the logs will be processed in a timely and efficient manner; and (ii) the cloud platform provides a continuous unified service by enabling redundancy for monitor resources and services, and mitigate the impact of single point of failure on the monitor service.

## 7 RELATED WORK

**Understanding the TLS/HTTPS ecosystem.** Several large-scale studies on the ecosystem have been finished. Holz et al. [41] analyzed the quality of TLS server certificates in the Internet, while Durumeric et al. [18] investigated the trust relationships among CAs and websites, and uncovered the insecure certificate practices. Amann et al. [5] analyzed the benign changes of the trust relationships in the wild. After analyzing about 47 million certificates, Perl et al. [64] found that only 66% of the 426 root CAs trusted on mainstream platforms are used to sign TLS server certificates. Huang et al. [42] investigated the MitM attacks against Facebook exploiting fraudulent certificates, and found that most were caused by antivirus software and corporate-scale content filters.

There are also reports on invalid certificates in practice. 65% of TLS certificates in the large-scale scan were invalid and most were held by end-user devices [11]. The study of the websites with invalid certificates shows that about one third of them configure invalid certificates accidentally, two thirds use invalid certificates deliberately [25]. Kumar et al. [45] analyzed the certificates in Censys and discussed the reasons of certificates with errors.

Based on the passive measurement of over 300,000 users over nine months, Akhawe et al. identified the low-risk TLS warnings that consume most user attention and proposed recommendations to browser developers that help to maintain the user attention in high-risk warnings [3]. The large-scale study in 2017 shows that most HTTPS errors are caused by client-side or network issues, instead of server misconfigurations [1].

The records in CT logs help to understand the TLS/HTTPS ecosystem. VanderSloot et al. [74] integrated the certificates in logs with the data from passive measurements, active scans, and certificate search engines, to present a complete view of the certificates in the wild. Using the data in public logs, Aertsen et al. [2] analyzed the certificate services of Let's Encrypt adopted in different organizations, hosts and domains, while Gasser et al. [26] investigated the violations of certificate issuance standards. With TLS server certificates from public logs and passive measurements, Cui et al. [15] analyzed the properties of forged certificates in the wild. Different from these studies using the data in CT logs to investigate certificate services, our work utilizes these data to analyze the services of CT monitors.

**CT deployment.** The deployments of CT in the Internet are investigated. Stark et al. [70] finished a comprehensive study on the adoption of CT across the web, including compliance, user experience, and potential risk. Gustafsson et al. [37] characterized 11 public logs with a focus on the recorded certificates and their usage in TLS/HTTPS. Nykvist et al. [58] studied the adoption of CT in Alexa Top-1M websites and the methods to deliver SCTs. Amann et al. [6] finished a large-scale study on the adoption of various TLS/HTTPS security enhancements, including CT, HPKP, HSTS, CAA, SCSV downgrade prevention and DANE. Scheitle et al. [67] analyzed the server-side deployment of CT, and discussed the domain name information leakage caused by the certificates in public logs. B. Li et al [48] explores the TLS/HTTPS configurations of third-party monitors, compared with common websites. These works studied the deployments of CT from the perspectives of logs and websites, while we focus on the implementation and deployment of reliable monitors.

**CT extensions and variations.** Following the basic CT framework, several designs were proposed to improve the security and/or performance of CT. Dowling et al. [17] defined the security properties of logging schemes, and formally proved that CT achieves these properties. An efficient gossip protocol was proposed to detect several types of log inconsistencies [10, 57]. The logs are audited without exposing user privacy by zero-knowledge proofs [21], and with the support of non-public subdomains by commitments with binding and hiding properties. Matsumoto et al. [49] studied the incentives to deploy the CT framework, and proposed the deployment status filters to detect the deployment status of a domain against the downgrade attacks.

The approach of CT is extended from certificate signing to other services. CIRT [66] records certificates in two Merkle hash trees: one of which is in chronological order with all certificates and the other is lexicographical with only the recent ones for each certificate subject, to achieve the transparencies of both certificate signing and revocation. Singh et al. [69] improved CIRT by bilinear-map accumulators and binary trees, to achieve the transparencies with shorter proofs. PKISN [71] records all certificates and revocations in public logs in chronological order. Then, it is able to revoke a CA certificate, while the end-entity certificates issued before the revocation do not become invalid. PoliCert [72] records subject certificate policies and certificates in public logs, providing the cryptographic proofs of presence and absence. CONIKS [50] builds transparent key directories based on Merkle prefix trees, allowing its users to audit their public keys while keep privacy. Software transparency [39] requires the developers to submit the updated package including all source codes and meta data, to public logs. Then, a monitor rebuilds all changed packages on every update and checks if the resulting binary matches. Inspired by the design of CT, the general transparency overlay [9] is proposed, which can be instantiated to provide transparency for other services (e.g., Bitcoin).

## 8 CONCLUSIONS

This paper presents the first systematic study on CT monitors in the wild. We analyze the data of 88 public logs in the Internet, and the studies show the (pre)certificates in the logs are increasing at a significant daily growth rate. The amount of records in the 50 regular logs increases by 7,778,870 records or 43.99 GB per day on average, and 5,002,599 records (about 30 GB) per day are appended in the 9 Google-operated & Google-approved logs since June 2018. The rapidly-increasing large amounts of records in the logs prevent an ordinary domain owner from assuming the role of monitor to watch for suspicious certificates by itself.

We study the certificate search services of well-known third-party monitors. Various combinations of domain names are input to search certificates from these monitors, and the search results disclose the following defects in the services: (*a*) none of the third-party monitors guarantees to return the complete set of certificates in the public logs for a domain; and (*b*) even the union of the third-party monitors can probably be unable to return the complete set, for some domains at least. The defective certificate search services of third-party monitors can cause some (fraudulent) certificates recorded in public logs but invisible to the claimed domain owners, which harms the overall reliability of CT.

This work provides an in-depth understanding of the implementations and deployments of CT monitors, and demonstrates several technical challenges of monitors in practice.

## ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their helpful comments. This research was supported in part by Cyber Security Program of National Key RD Plan of China (Grant No. 2017YFB0802 100), National Natural Science Foundation of China (Grant No. 61772518, 61572278 and U1736209), NSF DGE-1565570, NSA SoS Initiative and Ripple University Blockchain Research Initiative.

## REFERENCES

[1] Mustafa Emre Acer, Emily Stark, Adrienne Porter Felt, Sascha Fahl, Radhika Bhargava, Bhanu Dev, Matt Braithwaite, Ryan Sleevi, and Parisa Tabriz. 2017. Where the wild warnings are: Root causes of Chrome HTTPS certificate errors. In *14th ACM Conference on Computer and Communications Security (CCS)*. 1407–1420.

[2] Maarten Aertsen, Maciej Korczynski, Giovane Moura, Samaneh Tajalizadehkhoob, and Jan van den Berg. 2017. No domain left behind: Is Let's Encrypt democratizing encryption?. In *2nd Applied Networking Research Workshop (ANRW)*. 48–54.

[3] Devdatta Akhawe, Bernhard Amann, Matthias Vallentin, and Robin Sommer. 2013. Here's my cert, so trust me, maybe? Understanding TLS errors on the web. In *22nd International World Wide Web Conference (WWW)*. 59–70.

[4] Alexa. 2018. Alexa Top 1M Websites. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[5] Bernhard Amann, Robin Sommer, Matthias Vallentin, and Seth Hall. 2013. No attack necessary: The surprising dynamics of SSL trust relationships. In *29th Annual Computer Security Applications Conference (ACSAC)*. 179–188.

[6] Johanna Amann, Oliver Gasser, Quirin Scheitle, Lexi Brent, Georg Carle, and Ralph Holz. 2017. Mission accomplished? HTTPS security after DigiNotar. In *17th Internet Measurement Conference (IMC)*. 325–340.

[7] Apple Inc. 2018. Lists of available trusted root certificates in macOS. https://support.apple.com/en-us/HT202858.

[8] Apple Inc. 2019. Apple's certificate Transparency policy. https://support.apple.com/en-us/HT205280.

[9] Melissa Chase and Sarah Meiklejohn. 2016. Transparency overlays and applications. In *13th ACM Conference on Computer and Communications Security (CCS)*. 168–179.

[10] Laurent Chuat, Pawel Szalachowski, Adrian Perrig, Ben Laurie, and Eran Messeri. 2015. Efficient gossip protocols for verifying the consistency of Certificate logs. In *3rd IEEE Conference on Communications and Network Security (CNS)*. 415–423.

[11] Taejoong Chung, Yabing Liu, David R. Choffnes, Dave Levin, Bruce MacDowell Maggs, Alan Mislove, and Christo Wilson. 2016. Measuring and applying invalid SSL certificates: The silent majority. In *16th Internet Measurement Conference (IMC)*. 527–541.

[12] Comodo CA Limited. 2018. crt.sh: Certificate search. https://crt.sh.

[13] Comodo Group Inc. 2011. Comodo report of incident. https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html.

[14] David Cooper, Stefan Santesson, Stephen Farrell, Sharon Boeyen, Russell Housley, and W. Timothy Polk. 2008. IETF RFC 5280 - Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile.

[15] Mingxin Cui, Zigang Cao, and Gang Xiong. 2018. How is the forged certificates in the wild: Practice on large-scale SSL usage measurement and analysis. In *18th International Conference on Computational Science (ICCS)*. 654–667.

[16] Rasmus Dahlberg and Tobias Pulls. 2017. Verifiable Light-Weight Monitoring for Certificate Transparency Logs. arXiv ePrint. https://arxiv.org/abs/1711.03952.

[17] Benjamin Dowling, Felix Günther, Udyani Herath, and Douglas Stebila. 2016. Secure logging schemes and certificate transparency. In *21st European Symposium on Research in Computer Security (ESORICS)*. 140–158.

[18] Zakir Durumeric, James Kasten, Michael Bailey, and J. Alex Halderman. 2013. Analysis of the HTTPS certificate ecosystem. In *13th Internet Measurement Conference (IMC)*. 291–304.

[19] P. Eckersley. 2011. A Syrian man-in-the-middle attack against Facebook. https://www.eff.org/deeplinks/2011/05/syrian-man-middle-against-facebook.

[20] Graham Edgecombe. 2018. Certificate transparency monitor. https://ct.grahamedgecombe.com/.

[21] Saba Eskandarian, Eran Messeri, Joseph Bonneau, and Dan Boneh. 2017. Certificate transparency with privacy. In *17th International Symposium on Privacy Enhancing Technologies (PETS)*. 329–344.

[22] Chris Evans, Chris Palmer, and Ryan Sleevi. 2015. IETF RFC 7469 - Public key pinning extension for HTTP.

[23] Facebook Inc. 2018. Facebook: Certificate transparency monitoring. https://developers.facebook.com/tools/ct/search/.

[24] Facebook Inc. 2019. Facebook Monitor Statement. https://www.facebook.com/notes/protect-the-graph/detecting-phishing-domains-using-certificate-transparency/2037453483161459/, https://www.facebook.com/notes/protect-the-graph/introducing-our-certificate-transparency-monitoring-tool/1811919779048165/.

[25] S. Fahl, Y. Acar, H. Perl, and M. Smith. 2014. Why Eve and Mallory (also) love webmasters: A study on the root causes of SSL misconfigurations. In *9th ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*. 507–512.

[26] Oliver Gasser, Benjamin Hof, Max Helm, Maciej Korczynski, Ralph Holz, and Georg Carle. 2018. In log we trust: Revealing poor security practices with certificate transparency logs and Internet measurements. In *19th International Conference on Passive and Active Measurement (PAM)*. 173–185.

[27] GlobalSign. 2011. Security incident report. https://www.globalsign.com/resources/globalsign-security-incident-report.pdf.

[28] Google Inc. 2018. Certificate transparency. http://www.certificate-transparency.org/.

[29] Google Inc. 2018. Certificate transparency in Chrome. https://groups.google.com/a/chromium.org/forum/#!msg/ct-policy/wHILiYf31DE/iMFmpMEkAQAJ.

[30] Google Inc. 2018. Certificate transparency in Chrome policy. https://github.com/chromium/ct-policy/blob/master/ct_policy.md.

[31] Google Inc. 2018. Google: HTTPS encryption on the web. https://transparencyreport.google.com/https/certificates.

[32] Google Inc. 2018. Known logs. http://www.certificate-transparency.org/known-logs.

[33] Google Inc. 2019. Chromium STHSets. https://clients2.google.com/service/update2/crx.

[34] Google Inc. 2019. Google Gossip. https://www.gstatic.com/ct/gossip.

[35] Google Inc. 2019. Google Monitor Statement. https://transparencyreport.google.com/https/certificates.

[36] Google Inc. 2019. The root CA list accepted by the logs. https://groups.google.com/a/chromium.org/forum/#!msg/ct-policy/iNX-EmfRLOA/ctBCTYGfAwAJ, https://github.com/chromium/ct-policy/issues/.

[37] Josef Gustafsson, Gustaf Overier, Martin F. Arlitt, and Niklas Carlsson. 2017. A first look at the CT landscape: Certificate transparency logs in practice. In *18th International Conference on Passive and Active Measurement (PAM)*. 87–99.

[38] Heather Adkins. 2011. An update on attempted man-in-the-middle attacks. https://security.googleblog.com/2011/08/update-on-attempted-man-in-middle.html.

[39] Benjamin Hof and Georg Carle. 2017. Software distribution transparency and auditability. arXiv ePrint. https://arxiv.org/abs/1711.07278.

[40] P. Hoffman and J. Schlyter. 2012. IETF RFC 6698 - The DNS-based authentication of named entities (DANE) transport layer security (TLS) protocol: TLSA.

[41] Ralph Holz, Lothar Braun, Nils Kammenhuber, and Georg Carle. 2011. The SSL landscape: A thorough analysis of the X.509 PKI using active and passive measurements. In *11th Internet Measurement Conference (IMC)*. 427–444.

[42] Lin-Shung Huang, Alex Rice, Erling Ellingsen, and Collin Jackson. 2014. Analyzing forged SSL certificates in the wild. In *35th IEEE Symposium on Security and Privacy (S&P)*. 83–97.

[43] J. Kasten, E. Wustrow, and A. Halderman. 2013. CAge: Taming certificate authorities by inferring restricted scopes. In *17th Conference on Financial Cryptography and Data Security (FC)*. 329–337.

[44] Stephen Kent. 2019. IETF Draft - Attack and Threat Model for Certificate Transparency.

[45] Deepak Kumar, Zhengping Wang, Matthew Hyder, Joseph Dickinson, Gabrielle Beck, David Adrian, Joshua Mason, Zakir Durumeric, J. Alex Halderman, and Michael Bailey. 2018. Tracking certificate misissuance in the wild. In *39th IEEE Symposium on Security and Privacy (S&P)*. 785–798.

[46] Ben Laurie, Adam Langley, and Emilia Käsper. 2013. IETF RFC 6962 - Certificate transparency.

[47] Let's Encrypt. 2018. A free, automated, and open certificate authority (CA). https://letsencrypt.org/.

[48] Bingyu Li, Dawei Chu, Jingqiang Lin, Quanwei Cai, Congli Wang, and Lingjia Meng. 2019. The Weakest Link of Certificate Transparency: Exploring the TLS/HTTPS Configurations of Third-Party Monitors. In *18th IEEE International Conference On Trust, Security and Privacy in Computing and Communications (TrustCom)*.

[49] Stephanos Matsumoto, Pawel Szalachowski, and Adrian Perrig. 2015. Deployment challenges in log-based PKI enhancements. In *8th European Workshop on System Security (EuroSec)*. 1–7.

[50] Marcela S. Melara, Aaron Blankstein, Joseph Bonneau, Edward W. Felten, and Michael J. Freedman. 2015. CONIKS: Bringing key transparency to end users. In *24th USENIX Security Symposium*. 383–398.

[51] Microsoft Corporation. 2018. Certificate transparency in Microsoft. https://blogs.msdn.microsoft.com/azuresecurity/2018/04/25/certificate-transparency/.

[52] Microsoft Corporation. 2018. Microsoft trusted root certificate program: Participants. https://gallery.technet.microsoft.com/Trusted-Root-Program-d17011b8.

[53] B. Morton. 2014. More Google fraudulent certificates. https://www.entrust.com/google-fraudulent-certificates/.

[54] Mozilla. 2018. Mozilla included CA certificate list. https://wiki.mozilla.org/CA/Included_Certificates.

[55] Mozilla. 2019. Mozilla CT Policy. https://groups.google.com/a/chromium.org/forum/m/#!topic/ct-policy/Xx1bv8r33ZE.

[56] Nginx. 2018. Certificate transparency in Nginx. http://www.certificate-transparency.org/resources-for-site-owners/nginx.

[57] Linus Nordberg, Daniel Kahn Gillmor, and Tom Ritter. 2018. IETF Internet-Draft - Gossiping in CT. https://datatracker.ietf.org/doc/html/draft-ietf-trans-gossip-05.

[58] Carl Nykvist, Linus Sjöström, Josef Gustafsson, and Niklas Carlsson. 2018. Server-side adoption of certificate transparency. In *19th International Conference on Passive and Active Measurement (PAM)*. 186–199.

[59] OpenSSL. 2018. Certificate transparency in OpenSSL. https://www.openssl.org/docs/man1.1.0/crypto/ct.html.

[60] Opsmate Inc. 2018. Certificate transparency ecosystem. https://sslmate.com/labs/ct_ecosystem/ecosystem.html.

[61] Opsmate Inc. 2018. Certificate transparency log growth. https://sslmate.com/labs/ct_growth/.

[62] Opsmate Inc. 2018. SSLMate: Cert Spotter. https://sslmate.com/certspotter/.

[63] Opsmate Inc. 2019. SSLMate Statement. https://sslmate.com/blog/post/certspotter_apiv1, https://sslmate.com/certspotter/howithelps.

[64] H. Perl, S. Fahl, and M. Smith. 2014. You won't be needing these any more: On removing unused certificates from trust stores.. In *16th Conference on Financial Cryptography and Data Security (FC)*. 307–315.

[65] Erik Rogstad, Lionel Briand, and Richard Torkar. 2013. Test case selection for black-box regression testing of database applications. *Information and Software Technology* 55, 10 (2013), 1781–1795.

[66] Mark Dermot Ryan. 2014. Enhanced certificate transparency and end-to-end encrypted mail. In *21st ISOC Network and Distributed System Security Symposium (NDSS)*.

[67] Quirin Scheitle, Oliver Gasser, Theodor Nolte, Johanna Amann, Lexi Brent, Georg Carle, Ralph Holz, Thomas C. Schmidt, and Matthias Wählisch. 2018. The rise of certificate transparency and its implications on the Internet ecosystem. In *18th Internet Measurement Conference (IMC)*. 343–349.

[68] Patrick Schroeder, Pat Faherty, and Bogdan Korel. 2002. Generating expected results for automated black-box testing. In *17th IEEE International Conference on Automated Software Engineering (ASE)*. 139–148.

[69] Abhishek Singh, Binanda Sengupta, and Sushmita Ruj. 2017. Certificate transparency with enhancements and short proofs. In *22nd Australasian Conference on Information Security and Privacy (ACISP)*. 381–389.

[70] Emily Stark, Ryan Sleevi, Rijad Muminovic, Devon O'Brien, Eran Messeri, Adrienne Porter Felt, Brendan McMillion, and Parisa Tabriz. 2019. Does certificate transparency break the web? Measuring adoption and error rate. In *40th IEEE Symposium on Security and Privacy (S&P)*.

[71] Pawel Szalachowski, Laurent Chuat, and Adrian Perrig. 2016. PKI Safety Net (PKISN): Addressing the too-big-to-be-revoked problem of the TLS ecosystem. In *1st IEEE European Symposium on Security and Privacy (EuroS&P)*. 407–422.

[72] P. Szalachowski, S. Matsumoto, and A. Perrig. 2014. PoliCert: Secure and flexible TLS certificate management. In *21st ACM Conference on Computer and Communications Security (CCS)*. 406–417.

[73] University of Michigan. 2018. Censys. https://censys.io/.

[74] Benjamin VanderSloot, Johanna Amann, Matthew Bernhard, Zakir Durumeric, Michael Bailey, and J. Alex Halderman. 2016. Towards a complete view of the certificate ecosystem. In *16th Internet Measurement Conference (IMC)*. 543–549.

[75] VASCO Data Security International Inc. 2011. DigiNotar reports security incident. https://www.vasco.com/about-vasco/press/2011/news_diginotar_reports_security_incident.html.

[76] Dan Wendlandt, David G. Andersen, and Adrian Perrig. 2008. Perspectives: Improving SSH-style host authentication with multi-path probing. In *USENIX Annual Technical Conference (ATC)*.

[77] Wikipedia. 2017. Flame (malware). https://en.wikipedia.org/wiki/Flame_(malware).

[78] Wikipedia. 2018. List of Internet top-level domains. https://en.wikipedia.org/wiki/List_of_Internet_top-level_domains.

[79] K. Wilson. 2015. Distrusting new CNNIC certificates. https://blog.mozilla.org/security/2015/04/02/distrusting-new-cnnic-certificates/.

## APPENDIX

## A THE MONITORS' STATEMENTS ABOUT RELIABLE SERVICES

Several monitors under this study do claim to support services that return "*all of a domain's certificates that are record in active public CT logs*." Below are the quotations from their statements:

Google Monitor:[7] "*Use the search bar below to look up* ***all of a domain's certificates*** *that are present in active public certificate transparency logs. Site owners can search this site for domain names they control to ensure there have been no incorrect issuances of certificates referencing their domains.*"

SSLMate:[8] "*Reliable access to certificates: The Cert Spotter API* ***reliably returns all known, unexpired certificates for a domain name****, including those that were added to Certificate Transparency before you started monitoring but are not yet expired*;"

"*When picking a Certificate Transparency monitor to track your domain's SSL certificates, it's important to pick a monitor that won't miss certificates, even those that are very badly encoded*;"

"*Cert Spotter (SSLMate) tracks your organization's certificates to enhance your reliability and security.*"

Facebook Monitor:[9] "***Every time a new certificate appears in any public Certificate Transparency Log****, our tool analyzes the domains specified by the certificate for phishing attempts by taking into consideration the most common spoofing techniques - such as those described above. If it suspects that the domain is likely associated with phishing, it can notify subscribers of the tool for the legitimate*

[7] https://transparencyreport.google.com/https/certificates

[8] https://sslmate.com/blog/post/certspotter_apiv1;
https://sslmate.com/certspotter/howithelps

[9] https://www.facebook.com/notes/protect-the-graph/detecting-phishing-domains-using-certificate-transparency/2037453483161459/;

https://www.facebook.com/notes/protect-the-graph/introducing-our-certificate-transparency-monitoring-tool/1811919779048165/

**Table 10: The mainstream CAs unaccepted by public logs.**

| No.† | Subject | Validity Period | Platform |
|---|---|---|---|
| 1 | C=CN, O=China Financial Certification Authority, CN=CFCA Identity CA | 2015.06.30–2040.06.30 | Microsoft |
| 2 | C=US, O=Microsoft Corporation, CN=Microsoft Time Stamp Root Certificate Authority 2014 | 2014.10.22–2039.10.22 | Microsoft |
| 3 | C=US, O=Symantec Corporation, CN=Symantec Enterprise Mobile Root for Microsoft | 2012.03.15–2032.03.14 | Microsoft |
| 4 | L=Internet, O=VeriSign, Inc., OU=VeriSign Commercial Software Publishers CA | 1996.04.09–2004.01.07 | Microsoft |
| 5 | O=VeriSign Trust Network, OU=VeriSign, Inc., OU=VeriSign Time Stamping Service Root | 1997.05.12–2004.01.07 | Microsoft |
| 6 | C=AT, O=e-commerce monitoring GmbH, CN=GLOBALTRUST 2015 | 2015.06.11–2040.06.10 | Microsoft |
| 7 | C=Microsoft ECC Product Root Certificate Authority 2018, O=Microsoft Corporation, CN=US | 2018.02.27–2043.02.27 | Microsoft |
| 8 | C=ZA, O=Thawte, OU=Thawte Certification, CN=Thawte Timestamping CA | 1997.01.01–2020.12.31 | Microsoft |
| 9 | C=US, O=The USERTRUST Network, CN=UTN-USERFirst-Object | 1999.07.09–2019.07.09 | All |

†: These 9 mainstream CAs are not accepted by any Google-operated log, and the first 5 are not accepted by any regular log.

*domain by sending email, push, or on-site notifications, depending on their preference*;"

"*Using this freely accessible tool, you will now be able to receive alerts when a new certificate for any domain is added to the CT logs included in our database. Searching for certificates issued for a domain is also easy. Certificate Authorities issue hundreds of certificates every minute, but by using Facebook infrastructure, we can quickly process large amounts of data and* ***provide a reliable and efficient search function for certificates listed for a domain***."

## B DOMAIN NAMES WITHOUT CERTIFICATES

We have further investigated the 18 domains on the certificates deployed by the websites and why they were not returned by monitors. In summary, 10 websites support HTTPS but deploy their certificates in a problematic means. (*a*) Two websites (nextlnk1.com, nextlnk2.com) deploy certificates whose CNs do not match the domain names. While this does not violate any CT policy, it incurs difficulties for the monitors to process the certificates. (*b*) Two websites (blog.jp, pop.bid) deploy expired certificates (expired before our experiments on October 27, 2018), which are not covered in the experiments. (*c*) One website (freejobalert.com) deploys only a self-signed certificate, which has not been submitted to any CT log. And (*d*) five websites (lun.com, dytt8.net, igmatik.com, vseigru.net, seasonvar.ru) configure certificates which are issued after our experiments on October 27, 2018. The remaining 8 websites (haber7.com, bancodevenezuela.com, hdrezka.ag, livedoor.biz, mahresult.nic.in, doorblog.jp, iz682noju02ye5.com, hitpromoit.com) do not support HTTPS.

We believe the main reason that no certificate is returned for these 18 domains is due to the websites' own certificate deployment and HTTPS configuration, but not the monitors' defective processing. This is part of the reason that we define the reference set of certificates. The reference set contains all the known valid certificates, which are returned by at least one monitor, for domains under study.

## C MORE DETAILED EXPERIMENT RESULTS

Table 10 lists the mainstream CAs that are not supported by the public logs, and Table 11 gives the detailed certificate search results of 6,000 selected Alexa Top-1M websites.

**Table 11: The number of domains with incomplete results: groups of randomly-selected Alexa Top-1M websites.**

| | $\Phi$ | = 0 | = 1 | (1, 10] | (10, 100] | (100, +∞) | [0, +∞) |
|---|---|---|---|---|---|---|---|
| Top-1K | $D_\Phi$ | 18 | 50 | 235 | 471 | 226 | 1000 |
| | crt.sh | - | 0 | 8 | 28 | 68 | 104 |
| | SSLMate | - | 0 | 7 | 53 | 104 | 164 |
| | Censys | - | 0 | 3 | 33 | 84 | 120 |
| | Google Monitor | - | 0 | 59 | 269 | 218 | 546 |
| | Facebook Monitor | - | 0 | 14 | 102 | 173 | 289 |
| Top-(1K, 5K] | $D_\Phi$ | 32 | 82 | 309 | 452 | 125 | 1000 |
| | crt.sh | - | 0 | 12 | 35 | 31 | 78 |
| | SSLMate | - | 0 | 11 | 39 | 50 | 100 |
| | Censys | - | 0 | 3 | 16 | 33 | 52 |
| | Google Monitor | - | 0 | 57 | 245 | 119 | 421 |
| | Facebook Monitor | - | 0 | 21 | 190 | 96 | 307 |
| Top-(5K, 20K] | $D_\Phi$ | 62 | 134 | 310 | 423 | 71 | 1000 |
| | crt.sh | - | 0 | 4 | 17 | 25 | 46 |
| | SSLMate | - | 0 | 5 | 26 | 30 | 61 |
| | Censys | - | 0 | 1 | 9 | 17 | 27 |
| | Google Monitor | - | 1 | 55 | 175 | 63 | 294 |
| | Facebook Monitor | - | 4 | 69 | 257 | 63 | 393 |
| Top-(20K, 100K] | $D_\Phi$ | 122 | 179 | 377 | 297 | 25 | 1000 |
| | crt.sh | - | 0 | 10 | 15 | 4 | 29 |
| | SSLMate | - | 0 | 9 | 18 | 6 | 33 |
| | Censys | - | 0 | 5 | 5 | 4 | 14 |
| | Google Monitor | - | 1 | 62 | 113 | 22 | 198 |
| | Facebook Monitor | - | 8 | 68 | 164 | 19 | 259 |
| Top-(100K, 500K] | $D_\Phi$ | 166 | 218 | 390 | 222 | 4 | 1000 |
| | crt.sh | - | 0 | 6 | 10 | 0 | 16 |
| | SSLMate | - | 0 | 6 | 11 | 2 | 19 |
| | Censys | - | 0 | 4 | 2 | 1 | 7 |
| | Google Monitor | - | 2 | 51 | 61 | 3 | 117 |
| | Facebook Monitor | - | 12 | 85 | 125 | 4 | 226 |
| Top-(500K, 1M] | $D_\Phi$ | 228 | 233 | 392 | 143 | 4 | 1000 |
| | crt.sh | - | 0 | 3 | 8 | 0 | 11 |
| | SSLMate | - | 0 | 4 | 10 | 1 | 15 |
| | Censys | - | 0 | 3 | 3 | 1 | 7 |
| | Google Monitor | - | 1 | 33 | 36 | 3 | 73 |
| | Facebook Monitor | - | 9 | 70 | 77 | 4 | 160 |

$\Phi$: the range of the number of certificates for a domain.
$D_\Phi$: the number of domains in a group of websites. For every domain in this group, the sum of unique certificates returned is within the range $\Phi$.