Exploring the Evolution of TLS Certificates

Syed Muhammad Farhan and Taejoong Chung(✉)

Virginia Tech, Blacksburg, USA
{farhan,tijay}@vt.edu

Abstract. A vast majority of popular communication protocols such as HTTPS for the Internet employs the use of TLS (Transport Layer Security) to secure communication. As a result, there have been numerous efforts to improve the TLS certificate ecosystem such as Certificate Transparency logs and Free Automated CAs like LetsEncrypt. Our work highlights the effectiveness of these efforts using the Certificate Transparency logs as well as certificates collected via full IPv4 scans by validating them. We show that a large proportion of invalid certificates still exists and outline reasons why these certificates still exist. Additionally, we report unresolved security issues such as key sharing. Moreover, we show that the incorrect use of template certificates has led to incorrect SCTs being embedded in the certificates. Taken together, our results emphasize the continued involvement of the research community to improve the web's PKI ecosystem.

1 Introduction

TLS has become the de-facto standard for securing the Internet; it is the underlying security procedure behind popular communication protocols like HTTPS and SMTPS.

The widespread use of TLS has led to a lot of efforts from the community to make the TLS certificate ecosystem more democratic, transparent and economically feasible. Some efforts worth mentioning are the introduction of (1) ACME (specifically Let's Encrypt) [6] that allows valid certificates to be issued for free and removes the need for human intervention for certificate issuance and (2) the CT standard, which states that all compliant certificates must be published to append-only public servers so that any mis-issuance is promptly discovered, thus can be revoked.

This work presents an *audit* on the evolution of the certificate ecosystem over the last 8 years by using two sets of a large corpus of certificates; certificates collected from full IPv4 scans [20] from 2013 to 2021 and certificates logged in Google operated Certificate Transparency Logs till February 2021.

We make the following contributions. *First*, we explore how the overall *validity* of certificates has changed over time across most end-user applicable root stores. We observe that while the percentage of valid certificates has improved, a large portion of certificates are still invalid.

A. Brunstrom et al. (Eds.): PAM 2023, LNCS 13882, pp. 71–84, 2023.
https://doi.org/10.1007/978-3-031-28486-1_4

Second, we show that the use of template certificates to create new certificates has led to certificates presenting invalid extension data. Some certificates using template certificate fail to update key components in a certificates, which are supposed to be unique such as `subjectKeyIdentifier` and `ct_precert_scts` fields resulting in incorrect usage of these extensions.

Third, we show how the TLS certificate ecosystem has evolved over the past 8 years. We show that the overall security of certificates such as key strength has improved, and the ecosystem has become more centralized over time with a small number of CAs issuing a large percentage of total certificates.

2 Background

TLS Certificates: A TLS certificate binds a subject (domain) to a public key. These certificates are usually issued and signed by Certificate Authorities (CAs) once it successfully vets the subject. Thus, certificates usually have a certificate chain rooted in a widely-trusted set of root certificates, which are self-signed.

X.509 [12] is the most commonly used certificate management standard. X.509 certificates typically includes the subject (e.g., domain name), issuer (i.e., CA), public key, serial number (unique to a CA). It can also have additional information such as CRL Distribution Points extension [12], which allows a client to perform revocation check using URLs provided in the extension.

Invalid Certificates: A certificate can be *invalid* if it fails to meet certain rule in the RFC [12]; there can be multiple reasons that a client determines the certificate to be invalid; name a few, cryptographic errors (e.g., the signature of a certificate cannot be validated), expiration, self-signed certificates, etc.. A previous study [11] showed that the most common reason for this invalidity is certificates signed by untrusted root or self-signed and reported that 88% of invalid certificates are self-signed.

Certificate Transparency: Certificate Transparency logs [5] are public and append-only data structure, which are designed to ensure any certificate misissuance is caught early and can be revoked by the issuer. Over 6 billion certificates have been logged to Certificate Transparency logs to date [23].

Nowadays, Certificate Transparency sits within the wider ecosystem of the Web's PKI; CAs are expected to create a pre-certificate and log it to a CA when domain owners issue a Certificate Signing Request (CSR) to CAs. The CT, in return, sends an SCT (Signed Certificate Timestamp), which is a promise that the certificate will be added to the CT log within a predetermined timeframe. The CA then signs the final certificate and sends the certificate as well as the SCT to the domain owner. SCTs are usually embedded within a certificate, but may also be communicated through other means (e.g., an OCSP stapled response).

User agents (mostly browsers) validate SCTs when they receive a certificate to ensure that it has been logged to a Certificate Transparency log. Popular user agents have their own CT policy that determines how many and which CTs a certificate needs to be logged to be considered secure [10].

3 Related Works

Free and Automated CAs. While most measurement studies for PKI focus on valid certificates, a previous study [11] showed that a majority of certificates (88%) were invalid. The reason for these certificates being invalid was economical and a vast majority of invalid certificates originated from IoT devices. Since then, the introduction of Let's Encrypt [15] and the concept of free, automated CAs has made it increasingly easy and economically feasible to get valid certificates. Other CAs (like Sectigo [3] and cPanel [25]) soon followed suit and added support for automated certificate issuance.

Certificate Transparency. Benjamin et al. [7] explored different sources of SSL certificates and show that Certificate Transparency logs hold the largest collection of certificates. Still, CT was missing 15% of all certificates observed by the researchers; however, this research was conducted (in 2016) while Certificate Transparency was still in its infancy. Certificate Transparency became a standard industry practice and requirement for major browsers in 2018 as explained in [21]. Gasser et al. [14] focused on the syntactical compliance of certificates required by Baseline Requirements and found that nearly 900 k certificates are not compliant mainly due to a small number of CAs.

Korzhitskii et al. [16] characterized the root stores of popular Certificate Transparency logs and showed that while the CT root stores are expected to be a super-set of root stores from major browsers, most logs are missing a few root certificates from Apple, Microsoft or Mozilla root stores. Some studies [18,22] focused on the reliability and effectiveness of the CTlogs; for example, Stark et al. [22] measured the error rates and reported that it had been running with minimal breakage. However, Li et al. [18] showed that it is not practical to monitor the CT logs in a real time and process on top of them reliably due to the sheer volume of the CT logs.

4 Dataset and Methodology

4.1 Datasets

Our dataset consists of certificates collected via full IPv4 scans in project Sonar by Rapid7 [20] and the certificates logged to Certificate Transparency logs managed by Google.

IPv4 Scans: These scans are conducted by Rapid7, are open for public use, and are designed to find certificates from HTTPS endpoints. The timeline of this dataset spans from September 2013 to December 2021 with a total of 358,575,204 unique certificates observed. Scans were conducted every week from September 2013 to June 2017, every two weeks from June 2017 to January 2019, and daily afterwards. Additionally, from September 2013 to January 2018, only port 443 (the standard port for HTTPS) was scanned, while alternate HTTPS ports were

also scanned afterwards. We use the number of unique certificates as the unit of measurement for this dataset as a certificate is expected to be seen in multiple scans.

Certificate Transparency: Our Certificate Transparency dataset contains certificates published to CT logs operated by Google. We limit out CT dataset to logs managed by google since Chrome's CT policy requires certificates to be posted on at least one CT logs operated by google [10]. We collect a total of 4,481,716,844 certificates spanning from the inception of CT to February 2021. Since (1) IPv4 scans cannot fetch certificates that are only available through SNI (Server Name Indication) and (2) some certificates are not for Web (e.g., DANE [13]), we expect this is the cause of 12.5 times more certificates found in the CT logs.

The volume of certificates in CT was low before 2016, after which the number of certificates logged has been growing every year. We use the CT index as the unit of measurement for this dataset.

4.2 Validation Methodology

Root Stores. When validating certificates, the first question that needs to be answered is which root store should we use. Since our goal is to find out if end-user applications will find these certificates to be valid, we use four different root stores (Apple, Microsoft, Mozilla NSS, and Android root stores) to validate all certificates collected from IPv4 scan. Previously, Zane et al. [24] showed that a vast majority of root stores used in end user applications stem from a handful of 'root' root stores.

Since our certificate transparency dataset only includes certificates logged to Certificate Transparency logs managed by google, we want to use the root store from google's CT logs. Google states that the root stores for CT should be a super set of all major root stores (Apple, Microsoft and Mozilla) to be inclusive of all certificates that may be considered valid by these entities [4]. Looking at the current snapshot of root certificates for all the CT logs in our dataset, we find that the root stores for all active CT logs managed by google are the same. We then backtrack through the entirety of the timeline that CTs have been active to ensure we have all the historical root certificates in our root store.

Validation Configuration. We use the command line version of OpenSSL to validate all certificates in our dataset and set it up to ignore time related errors and certificate revocations.

To validate certificates collected via IPv4 scans, we first isolate all CA certificates (certificates with the is_ca tag set to true) and iteratively validate them. After each iteration, we add valid CA certificates to our set of possible intermediate certificates and repeat this process until there are no new valid CA certificates. Finally, we verify all the leaf certificates using the full set of intermediate certificates.

The rules governing which certificates may be added to CT are more relaxed than those generally used to validate x509 certificates where inclusion to CT only

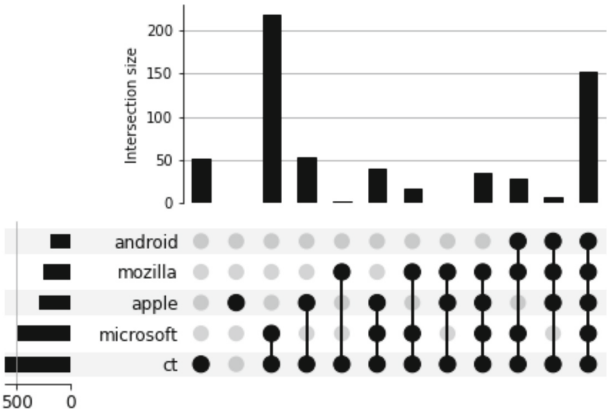

Fig. 1. Set relationship between root stores.

requires a valid chain to a root certificate [17]. To validate CT certificates, we setup OpenSSL to ignore x509 critical errors (CT pre-certificates have a poison added to make them invalid for regular use) and use the intermediate certificates logged to CT.

5 Certificate Validity

5.1 IPv4 Scanning

Unless stated otherwise, we define a certificate as valid if it is valid for any end-user root store in our dataset. After validating all certificates in our dataset, we isolate 121,062,606 unique valid certificates (33.76% of all certificates). We observe that the majority (66.23%) of certificates are *invalid* across all the root stores we use in our test; more specifically, 55.41% (131,625,055) of the invalid certificates are invalid because they are self-signed and 44.58% (105,770,334) are invalid because they are signed by another invalid certificate. This accounts for the vast majority of invalid certificates with only 1074 certificates invalid due to some other reason.

Figure 1 shows the set relationship across different root stores. We find that the CT root store contains all the root certificates from other root stores apart from one certificate that is only present in the Apple root store. We also observe that while a large number of certificates are shared between all end-application root stores, a large number of certificates are unique to the Microsoft and Apple root stores. However, there is little variation in the validity of certificates between different root stores. We find that 120,200,897 certificates (99% of the certificates valid in any root store) are valid across all end-application root stores in out dataset. This echoes the work done by Pearet al. showing that a vast majority of HTTPS servers use CAs that are trusted by all major trust stores [19].

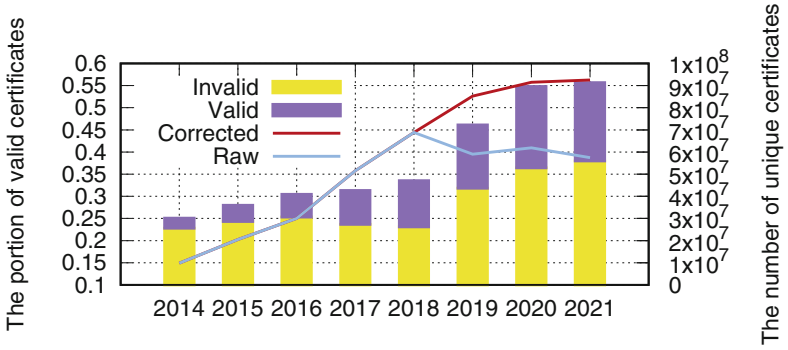

Fig. 2. The number of valid and invalid certificates via IPv4 scanning

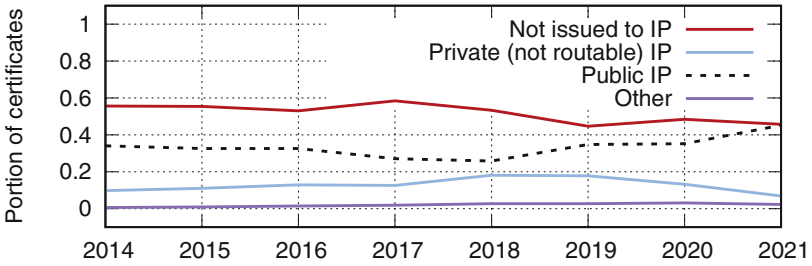

Fig. 3. The number of invalid certificates issued to IP addresses

The total number of certificates and percentage of valid certificates has been on the rise throughout our scanning period as shown in Fig. 2. We surprisingly saw a declining validity percentage after 2018, which we later found was an artifact of more frequent scanning since a large percentage of invalid certificates are seen in a single scan and are ephemeral. The corrected validity percentage (where we sample our data after 2019 to be consistent with the prior data) shows a constant increase over time. However, there remains a large percentage (45%) of invalid certificates.

Why are there still invalid certificates? Almost half of the invalid certificates are issued to IP addresses (i.e. the common name is an IP address) as highlighted in Fig. 3 (the other half has a valid FQDN); we find that the percentage of invalid certificates issued to private IPs has been on the rise for the past three years, while the percentage of invalid certificates not issued to IP addresses has been on the decline. Only a minute number of valid certificates are issued to IP addresses as CAs will generally not issue certificates without a valid domain name.

Where are Invalid Certificates Hosted? We use the common name of the certificate and the subject alternative names of the certificates to find all the domains a certificate represents. Note that some certificates are issued to IP addresses and are excluded from this analysis (and some certificates counted multiple times in different domains) Fig. 4 shows the top 5 top level domains in

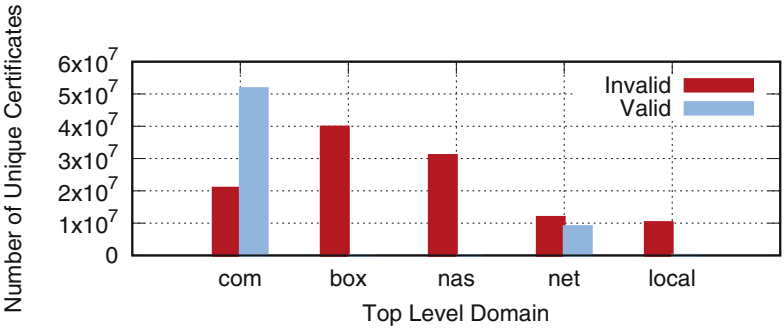

Fig. 4. Popular top level domains for valid and invalid certificates

our dataset. We observe that popular web domains like .com and .net hosting publicly accessible web pages tend to present valid certificates. Throughout the manual investigation, we find that invalid certificates for the .net domain are largely issued by Kubernetes. The .box and .nas domains (over 99.9%) are almost exclusively invalid and routers from AVM (fritz box) constitute a large majority of these certificates; thus, we believe that such invalid certificates are used for individual applications (such as using a network attached storage device). Since the .local domain is reserved for use by Internet Engineering Task Force (IETF), thus we exclusively observed them in invalid certificates. We could not identify any patterns that might tell us their source.

5.2 Certificate Transparency

Validating the certificates in the CT logs, we find that only few certificates are invalid as shown in Fig. 5. Some of these invalid certificates were added to the log due to bugs in the CT code; for example, in 2015, a bug introduced to the Google Pilot and Aviator logs accepted all certificates with unsupported algorithms [2].[1] Interestingly, these logs were not removed from the trusted set for Google Chrome. As the set of root certificates used by a CT log is arbitrary, any certificate may be added to the CT log without any consequence since any user agent will have their own root store and validate the certificate in question. The reason for the filter in Certificate Transparency is due to operational reasons like reducing the amount of spam in the logs, and keeping the log servers available at all times.

6 Certificate Authorities

6.1 IPv4 Scanning

Figure 6 tracks how domains (CNs) have been migrated across different CAs. We can infer that common names representing invalid certificates tend to be short-lived, since the percentage of invalid certificates in Fig. 6 does not match the one

[1] It is worth noting that these problematic certificates cannot be removed because the CT log is a Merkle-tree based structure, which is append-only.

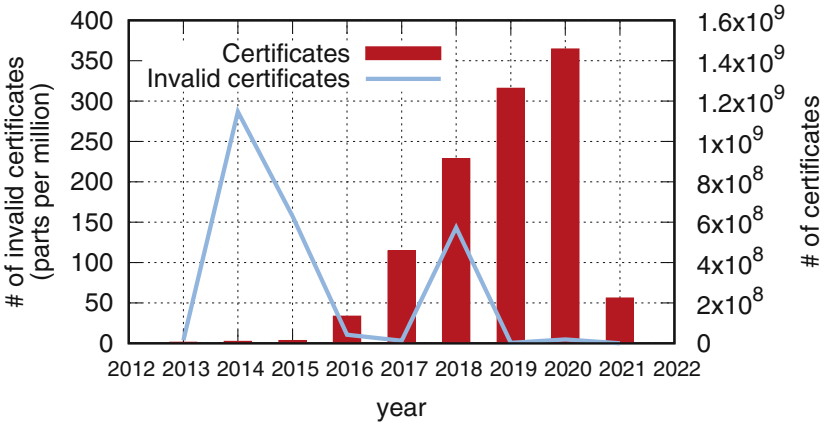

Fig. 5. The number of total (and invalid) certificates in the CT logs

in Fig. 2. We observe the growth of Let's Encrypt, which issued an insignificant amount of certificates in 2015, which is the largest issuer of certificates in 2021. Note that most domains that use Let's Encrypt and are new TLS (either new to using TLS or new domains altogether) as highlighted by the fact that we cannot find certificates for these domains before the one issued by Let's Encrypt. The increase in flow for Digicert is explained by the fact that Digicert acquired Symantec in October 2017. The decrease in flow for Comodo CA Limited between 2018 and 2019 is because it was rebranded as Sectigo in November 2018. Fortinet had a small presence up till 2017, after which we see a sharp increase in the number of certificates issued by Fortinet. This is likely an artifact of scanning since most of the Fortinet certificates are found on port 8010 and our dataset did not include this port before 2018.

Overall, centrality of issuers has increased over time where only a small percentage of certificates were issued by top CAs in 2013, but a large fraction of certificates are issued by Let's Encrypt in 2021. Additionally, we also find that only 10 keys are used to directly sign 80% of the valid certificates, which brings a security concern; if any of these keys are compromised, there will be a disproportionately large impact on the health of the ecosystem.

6.2 Certificate Transparency

We find that Let's Encrypt dominates the Certificate Transparency logs, consistently issuing 80% of the certificates logged after 2016; Note that the volume in CT was quite low prior to 2016. As shown in Fig. 7, cPanel and Sectigo also consistently log a substantial portion of certificates. Since the vast majority of the certificates from CT Logs are valid as shown in Fig. 5, we could not find any distinctive pattern in terms of the population of invalid certificates acrossthe CAs.

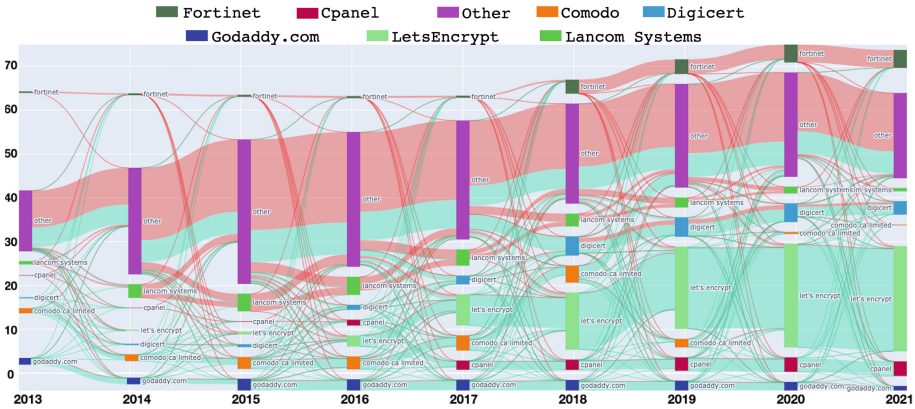

Fig. 6. Tracking yearly CA choice for all common names in the certificates from IPv4 scanning. The red flow represents invalid certificates while the blue flow represents valid certificates. (Color figure online)

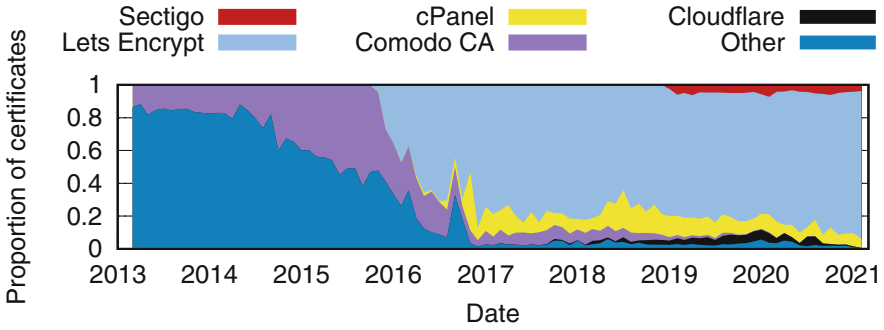

Fig. 7. Proportion of certificates logged to CTs by CA

7 Host Networks

Now, we focus on where the certificates are hosted from by looking at the IP address of hosts that serve certificates.

7.1 IPv4 Scanning

Similar to previous approach [11], we use CAIDA AS classification dataset [8] and group the certificates by the type of AS. Figure 8 shows the distribution of AS types over time for both valid and invalid certificates; we first immediately notice that invalid certificates can predominately be found on transit/access type ASes, which correspond to end-user connections; however, the portion of such certificates decreases as time goes on.

Valid certificates, on the other hand, are much less likely to be found on transit addresses. We also see a decreasing trend in the usage of transit addresses

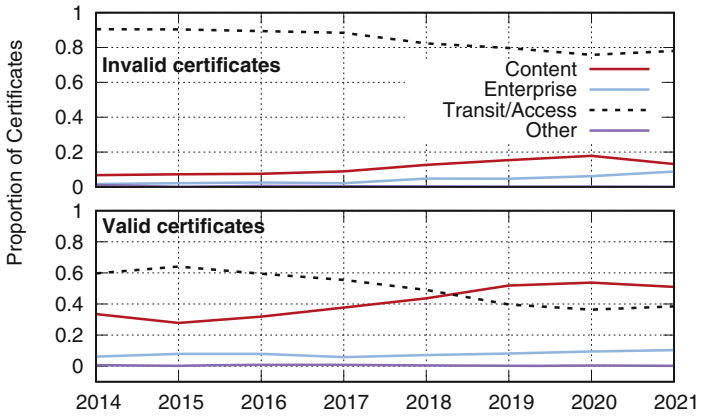

Fig. 8. Distribution of AS types over time for certificates collected through IP scanning

for valid certificates while the proportion of valid certificates hosted via content ASes is increasing.

7.2 Certificate Transparency

For the certificates collected from September 2020 to March 2022, we run DNS queries on the domain names for which certificates are posted on CT to get the A records mapped to the domain. We then follow the procedure highlighted above to get the AS number, and show the results in Fig. 9. Comparing with Fig. 8, we see a larger portion of certificates logged to CT hosted on content ASes as compared to valid certificates seen in IP scans for the same time range; this is aligned with a previous study [9], which showed that many websites are hosted at least in part by third parties, more centralized in CDNs.

8 Evolution

This section describes the major changes we observe in the IP scanned certificates.

Signature Algorithm: In early 2017, major browsers including Chrome, Firefox and Safari officially depreciated the use of SHA1 as an encryption algorithm for certificates [1] as there are significant collision attacks available for the algorithm. Almost all valid certificates shifted to using SHA256 in favor of SHA1 by 2017. The same change happened much slower for invalid certificates with a considerable portion of certificates still employing SHA1 as late as 2020.

Certificate Revocation. We rarely find a certificate revocation mechanism defined for invalid certificates. For valid certificates, we find that CRLs were very popular till 2015, after which we observe a constant decline in the percentage of valid certificates supporting CRLs. OCSP was quite popular in 2013 with 90% of

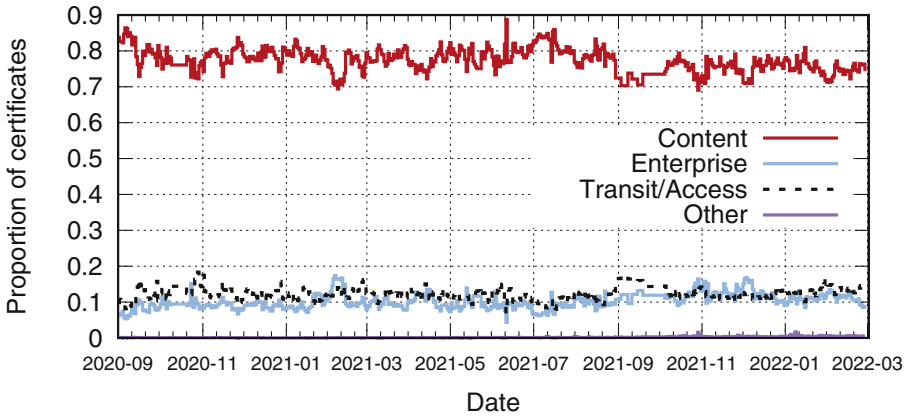

Fig. 9. AS types of certificates logged to CT logs

the valid certificates supporting it, but after 2015 practically all valid certificates support OCSP.

CT Inclusion Proofs. Figure 10 shows the inclusion in CT for Rapid7 scanned certificates over time. More than 99% of valid certificates after 2018 are included in CT logs, while a small but increasing proportion of invalid certificates have SCTs. While the rules governing Certificate Transparencies to filter certificates are more lenient than those for validity, it is unexpected to find invalid certificates with SCTs. We find that these certificates are sharing SCTs. SCTs should be cryptographically generated by the CT log and thus valid SCTs must be unique. Only 24% of invalid certificates have a unique SCT, in contrast we find no valid certificate sharing an SCT with another valid certificate. We revalidate these certificates (the ones that have unique SCTs) using the CT root store and corresponding rules and find that a majority of these may be considered valid by CT [17].

99% of the invalid certificates sharing SCTs are issued by Fortinet. In the general case we find one valid certificate who's SCT is shared by multiple invalid certificates. Moreover, other (certificate specific) extensions, like Subject Key Identifier are shared among these certificates even though they do not share a public key. We believe that these are a result of using existing valid certificates as a template to create new certificates. This explains why these certificates present invalid SCT tags and share the Subject Key Identifiers. We reached out to Fortinet for a comment but have not received a response.

9 Discussion

Free, Automated CAs like Let's Encrypt and cPanel have been the biggest source of change for the TLS certificate ecosystem. These services allow valid certificates to be issued without any real investment from the domain owner (both in terms

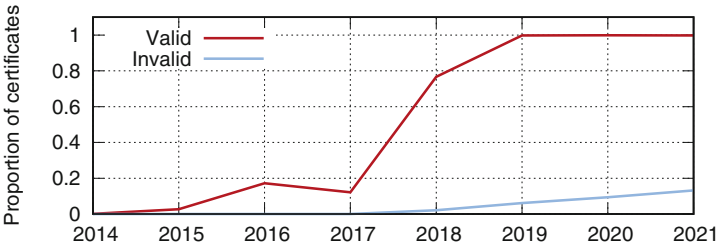

Fig. 10. Proportion of valid and invalid certificates with SCTs defined

of time and money). The improvement in validity percentage that we see is also in large part due to these Certificate Authorities. We can also attribute the decrease in validity period to these CAs.

On the other hand, these ACME-supporting CAs also put the overall ecosystem at risk. The nature of the PKI ecosystem means that any domain can be impersonated by compromising the least secure CA, and there are known attacks against the domain validation employed by these CAs. We observe that centrality in issuers has increased considerably over the course of our scans, and this is likely caused by the popularity of Let's Encrypt and cPanel. Only 10 keys are responsible for directly signing 80% of the valid certificates, which means that in case these keys are compromised, a mast majority of valid certificates should become invalid as the certificates with these keys are revoked, or are removed from trust stores. These keys are likely to be compromised when compared to keys used for root certificates, as the root certificate keys are rarely held in memory to sign other certificates, while these are continuously held in memory.

10 Conclusion

This work presents a bird's eye view of how the web's PKI ecosystem has evolved over the past 8 years. The validity of certificates has improved consistently, but a large proportion of certificates are still invalid. Over time, most indicators show that the ecosystem is moving towards better security practices. However, there are a few alarming trends including the incorrect use of template certificates causing invalid extensions and increasing centrality in issuers.

Acknowledgments. We thank the anonymous reviewers and our shepherd, Oliver Gasser, for their helpful comments. We also thank Christo Wilson and Alan Mislove for allowing us to use their computing infrastructure and Zane Ma for sharing datasets and discussion. This research was supported in part by NSF grants CNS-2053363 and CNS-2051166.

A Ethics

This paper does not pose any ethical issues as all the data we use for analysis is collected by third parties and is available for public use.

References

1. Browser security icon updates And sha-1 deprecation — digicert. Com. https://www.digicert.com/blog/browser-security-icon-updates-sha-1-deprecation
2. incident response: november 2015 google 'pilot' And 'aviator' Logged 3 Certs With Invalid Signatures
3. Sectigo Adds Acme Protocol Support In Certificate Manager Platform To Automate Ssl Lifecycle Management. 2019. https://sectigo.com/resource-library/sectigo-adds-acme-protocol-support-in-certificate-manager-platform-to-automate-ssl-lifecycle-management
4. Certificate transparency Website. Google, March 2022. https://github.com/google/certificate-transparency-community-site/blob/4ae90f78cdd821b9cb3a6884885186d71b529c87/docs/google/getting-started.md
5. Laurie, et al. Certificate Transparency. RFC 6962 (Proposed Standard), IETF, June 2013
6. Barnes, R., Hoffman-Andrews, J., McCarney, D., Kasten, J.: Automatic Certificate Management Environment (acme). IETF, March 2019
7. Sloot, V., et al.: Towards a complete view of the certificate ecosystem. In: Proceedings of the 2016 Internet Measurement Conference, New York, NY, USA, pp. 543–549, Association for Computing Machinery (2016)
8. Caida asclassifications Dataset. https://www.caida.org/data/as-classification/
9. Cangialosi, F.,et al.: Measurement and analysis of private key sharing in the https ecosystem. In: ACM Conference on Computer and Communications Security (CCS), Vienna, Austria, October 2016
10. Certificate Transparency In Chrome (2019). https://github.com/chromium/ct-policy/blob/master/ct_policy.md
11. Chung, T., et al.: Measuring and applying invalid SSL certificates: the silent majority. In: ACM Internet Measurement Conference (IMC), Santa Monica, California, USA, November 2016
12. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. RFC 5280, IETF, May 2008. https://www.ietf.org/rfc/rfc5280.txt
13. Dukhovni, V., Hardaker, W.: The DNS-based authentication of named entities (DANE) protocol: updates and operational guidance. RFC 7671, IETF, October 2015
14. Gasser, O., Hof, B., Helm, M., Korczynski, M., Holz, R., Carle, G.: In log we trust: revealing poor security practices with certificate transparency logs and internet measurements. In: Passive and Active Measurement Conference (PAM) (2018)
15. Aas, J., et al.: Let's encrypt: an automated certificate authority to encrypt the entire web. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, New York, NY, USA, pp. 2473–2487, Association for Computing Machinery (2019)
16. Korzhitskii, N., Carlsson, N.: Characterizing the root landscape of certificate transparency logs. In: 2020 IFIP Networking Conference, Networking 2020, Paris, France, 22–26 June 2020, pp. 190–198. IEEE (2020)
17. Laurie, B., Langley, A., Kasper, E.: Certificate Transparency. RFC 6962, IETF, June 2013. https://www.ietf.org/rfc/rfc6962.txt
18. Li, B., et al.: Certificate transparency in the wild: exploring the reliability of monitors. In: ACM Conference on Computer and Communications Security (CCS) (2019)

19. Perl, H., Fahl, S., Smith, M.: You won't be needing these any more: on removing unused certificates from trust stores. Financial Cryptography and Data Security (FC), Christ Church, Barbados, March 2014
20. Project sonar. https://www.rapid7.com/research/project-sonar/
21. Scheitle, Q., et al.: The rise of certificate transparency and its implications on the internet ecosystem. In: Proceedings of the Internet Measurement Conference 2018, New York, NY, USA, pp. 343–349. Association for Computing Machinery (2018)
22. Stark, E., et al.: Does certificate transparency break the web? Measuring adoption and error rate. In: IEEE Symposium on Security and Privacy (IEEE S&P) (2019)
23. Working Together To Detect Maliciously Or Mistakenly Issued Certificates. https://certificate.transparency.dev/
24. Ma, Z., Austgen, J., Mason, J., Durumeric, Z., Bailey, M.: Tracing your roots: exploring the TLS trust anchor ecosystem. In: Proceedings of the 21st ACM Internet Measurement Conference, New York, NY, USA, pp. 179–194, Association for Computing Machinery (2021)
25. Cpanel Auto SSL. https://docs.cpanel.net/knowledge-base/security/guide-to-ssl/#autossl