TLS Certificate Analysis

Jonas Lang, Markus Sosnowski*, Johannes Zirngibl*, Patrick Sattler*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: langj@cs.tum.de, sosnowski@net.in.tum.de, zirngibler@net.in.tum.de, sattler@net.in.tum.de

***Abstract*—TLS Certificates contain a variety of different information. In this paper, we give an overview on what information we could extract from a large dataset of TLS certificates. We gather information about who issues these certificates and what they do with select fields. Also, we use zLint, a linter, to check certificates for issues. Finally, we look at the trust chains of leading to certificates.**

***Index Terms*—tls certificate analysis, internet-wide, zlint, trustchain, PKI**

1. Introduction

In recent years, using HTTP over TLS (HTTPS) has become more and more widespread for securely communicating over the web. Major Web-Browsers are marking websites using the plain Hypertext Transfer Protocol (HTTP) as "insecure" [1], [2], so websites are incentivized to provide access by HTTPS [3]. While establishing a Transport Layer Security (TLS) connection, the server has to present a X.509 Certificate. This is also called a TLS certificate. The client has to determine if the certificate is valid, and if the certificate can be trusted. If the certificate is considered invalid or not trusted, web-browsers may block access and display a warning [4], [5]. In 2019, the management of TLS certificates has been standardized by the Automatic Certificate Management Environment (ACME) protocol. This protocol simplifies automated issuance, renewal and revocation of a certificate [6]. In this paper, we focus on the analysis of TLS certificates. We first introduce the concept of TLS certificate in section 2". Then we present a short overview of the design of our analysis in section 3", followed up by the details of the implementation in section 4". The core part of our paper is the evaluation of the results in section 5". Finally, we draw a conclusion and present opportunities for future work in section 6".

1.1. Related Work

There have been other papers that surveyed the certificates used for TLS. The authors of "Analysis of the HTTPS Certificate Ecosystem" [7] presented in 2013 a large-scale study that gave insight into the HTTPS certificate ecosystem. They analysed over 42M certificates in total, and investigated the trust relationships between users, intermediate authorities and root authorities. Another Study in 2017 looked at the misissuance of certificates. "Tracking Certificate Misissuance in the Wild" [8] introduces zLint, a certificate linter. They have been able to check 61M certificates for misissuance and uncovered that mainly smaller organizations misissue certificates.

2. Background

TLS certificates are mainly used by servers to authenticate themselves. Most certificates are leaf certificates, that are signed by a 3rd Party, called a certificate authority (CA). The CAs control root certificates, which are the trust anchor in this system - most clients trust a set of root certificates, transitively trusting each certificate signed by one of the roots. However, most leaf certificates are not directly signed by root certificates, instead they are signed by intermediate certificates. Those intermediate certificates are signed by root certificates. TLS certificates can also be self-signed, therefore they are not signed using the private key from a third party. This means that the client needs to trust the certificate. Most browsers offer the option to trust certain self signed certificates, but self-signed certificates are generally not stored in the trust anchor.

2.1. Chain of Trust

If a server presents a leaf certificate to a client, the client has to determine if it trusts this certificate. As the client implicitly trusts all root certificates, the client has to build a chain of trust to a root certificate, as depicted in Figure 1. In other words, the client has to find a root certificate that has signed the presented leaf certificate directly, or a chain of potentially multiple intermediate certificates that lead to the leaf certificate.

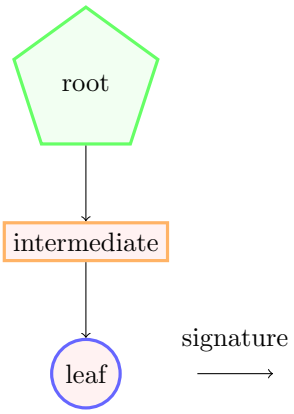

Figure 1: chain of trust

doi: 10.2313/NET-2021-05-1_10

2.2. Certificates with more than one Parent

It is possible that there are two or more valid chains of trust associated with the same leaf certificate. E.g. this occurs when two intermediate certificates share the same private key so both their signatures are identical, as in Figure 2.

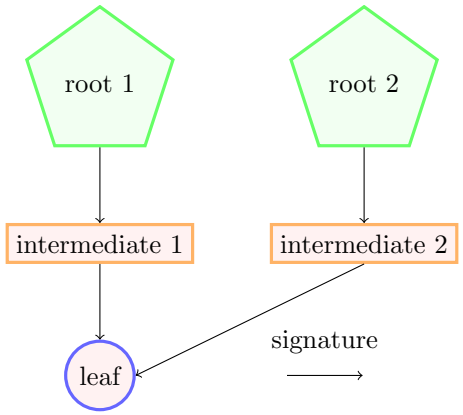

Figure 2: example certificate with two valid chains of trust

2.3. Mississued Certificates

Misissued certificates do not adhere to specifications in RFC5280 [9], or fail to adhere to the CA/Browser Forum Baseline Requirements [10]. If certificate fails to meet those requirements, a client typically does not trust the server.

3. Design

We use a large-scale HTTPS scan on port 443 created in December 2020 using goscanner [11] containing over 126M TLS certificates. The certificates are parsed, processed and then results are written back to disk to be analyzed. We examine the presence of select fields and analyse their content. Each certificate is checked by a certificate linter. Furthermore, we use a library to build trust chains for each certificate, that end with a root certificate in our trust anchor.

4. Implementation

Each certificate is parsed using the zCrypto library, which is based on the standard Go library [12]. If the Basic Constraints extension is present and the cA boolean [sic] is set, a a certificate is considered a CA certificate, otherwise, it is considered a leaf certificate [9].

4.1. Linting with zLint

Every certificate was linted by running zLint on it, which checks for "consistency with rfc standards and other relevant pki requirements" [13]. zlint distinguishes three categories of Lints: "Notice", "Warn" and "Error". Lints in category "Notice" can be non-deterministic and indicate there may be a problem. As over 126M certificates were processed in this paper, it was not possible to examine for every notice if there truly was a problem, so lints of this category were ignored. Lints in category "Warn" check e.g. if a SHOULD or SHOULD NOT Requirement from an RFC has been violated, while lints in category "Error" e.g. check if a MUST or MUST NOT Requirement has been violated [13].

4.2. Building trust chains

We use the mozilla root store [14] as of December 2020 as the trust anchor. The set of intermediate certificates is built by collecting all certificates that are CA certificates. Then we use the Verify function from zCrypto [15] to find all trust chains that lead to a root in the trust anchor.

4.3. Differences to typical clients in trust chain validation

Some certificates that appear valid in our testing may be rejected by certain browsers, and some certificates that appear valid in certain browsers may appear valid in our testing.

4.3.1. Available Intermediates. We try to build trust chains using the set of all intermediate certificates that have been seen in the scan. Typically however, a client should rely on the server to present all intermediate certificates leading to the root. All clients we are aware of use some form of caching for intermediate certificates. If the necessary intermediates are already in this cache, the client may succeed in building a trust chain even if the server does not present every necessary intermediate. Some clients try to use an Uniform Resource Identifier (URI) specified in the Authority Information Access Extension (AIA Extension) [15] to fetch missing intermediate certificates. The platform verifiers on Windows, ChromeOS and MacOS implement this. [16] However, a major client that does not support fetching intermediate certificates using the AIA Extension is Mozilla Firefox [17].

4.3.2. Revoked certificates. In theory, CAs should be able to revoke issued certificates via the Online Certificate Status Protocol (OCSP) and Certificate Revocation Lists (CRL). As Liu et. al found in 2015 however, these mechanisms are often not used by clients [18]. They also revealed that Mobile Browsers on Android did not check for revocation at all. Some Desktop-Browsers use preselected CRLs to check for revoked certificates, which only contain a fraction of all revoked certficates [19], [20]. Because the revocation of certificates is handled so differently across clients, we do not check any certificates for revocation.

5. Evaluation

We ran our analysis on a Dataset that was collected from 24/12/2020 to 27/12/2020. This Dataset contains 126.200.987 certificates that could be successfully parsed, while only 175 certificates were too broken to be parsed.

TABLE 1: Classes of certificates in our dataset

class	total %	absolute
all certificates	100%	126.200.987
roots	<0,01%	78
intermediates	0,18%	236.475
valid intermediates	<0,01%	927
self-signed	1,75%	2.207.833
leaves	98,24%	123.983.769
valid leaves	95,61%	120.658.574

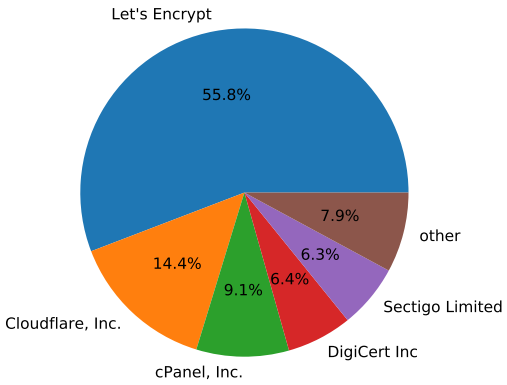

Figure 3: Percentage of valid leaves signed by each organization

5.1. General Landscape

We define a valid leaf as a leaf certificate, to which we could build a trust chain that has been valid anywhere in between 21/12/2020 and 28/12/2020. This is to mitigate that the certificates were collected over multiple days. Table 1 reveals that over 95% of certificates are valid leaf certificates.

5.2. Issuers of HTTPS Certificates

We grouped valid leaf certificate by issuer organisation name in Figure 3 . Let's Encrypt is the dominant issuer of TLS certificates with a share of over 55%.

5.3. Signature Algorithms

Table 2 shows that certificate signatures among valid leaves were almost exclusively made using SHA256-RSA. An exception is Cloudflare, which is responsible for 99,55% of all ECDSA-SHA256 signatures, which amounts to 17.405.658 certificates. This is 99,84% of all Cloudflare issued certificates.

TABLE 2: Signature Algorithms used for valid leaves

algorithm	valid leaves %	absolute
SHA256-RSA	85,23%	102.834.585
ECDSA-SHA256	14,49%	17.483.284
SHA384-RSA	0,27%	316.893
ECDSA-SHA384	0,01%	17.483.284
SHA512-RSA	<0,01%	6.278

TABLE 3: Certificate lifespan by interval

lifetime in days	valid leaves %	absolute
51-100	70.80%	85.427.923
200-398	26.80%	32.341.146
399-800	2,00%	2.413.525
801-1200	0,23%	282.170
101-200	0,11%	132.037
0-50	0,05%	61.773

TABLE 4: Usage of the subject country field

country	valid leaves %	absolute
not used	82,76%	104.443.306
US	14,79%	18.336.360
PL	0,12%	156.190
DE	0,11%	134.287
...	...	...

5.4. Validity Period

Each certificate has a not before and not after field. These indicate the lifespan of a certificate. In Table 3, the lifetime of valid leaf certificates is divided into classes. Starting on September 1st 2020, every major root program decided to reduce the maximum lifespan of each certificate to 398 days [21]. The majority of valid leaves has a considerably shorter validity period of <100 days. This is desirable, as shorter lifespans of certificates are generally better for security, as certificates have to be reissued more often. Certificates that have a longer lifespan than 398 days, but have been issued before September 2020 can still be valid certificates.

5.5. Subject Country Field

As X.509 certificates used for TLS are general purpose, they also feature fields that are not useful for typical browsers. One example is the subject country field, which is left unused by the vast majority of certificates, as Table 4 shows. Surprisingly, some CAs like Cloudflare seem to set the subject country field for all of their issued certificates to "US".

5.6. Extended Key Usage Field

Over 98% of certificates are issued with ServerAuth and ClientAuth as their extended key usage, as Table 5 shows. The purpose of the certificates during the scan was to authenticate the server, so ClientAuth should not be needed as specified by RFC5280, Section 4.2.1.3. [9]. We hypothesize that the vast number of certificates with ClientAuth set is due to the added flexibility for the users.

TABLE 5: Extended key usage

Extended key usage	valid leaves %	absolute
ServerAuth, ClientAuth	98,15%	118.407.285
ServerAuth	1 ,84%	2.243.899
...	...	...

doi: 10.2313/NET-2021-05-1_10

TABLE 6: certificates with warnings per organization sorted by absolute amount

Organisation Name	warnings	issued	% warnings
GoDaddy.com, Inc.	179.679	2.481.067	7,24%
Amazon	40.853	1.392.205	2,93%
SECOM Trust Systems [...]	35.842	35.872	99,91%
Actalis S.p.A.	20.984	408.090	5,14%
DigiCert Inc	20.389	7.773.648	0,26%
Starfield Technologies, Inc.	17.305	322.869	5,35%
Microsoft Corporation	13.572	123.668	10,97%
Sectigo Limited	9.399	7.588.983	0,12%
...	...	...	...

TABLE 7: valid leaf certificates with errors per organization sorted by absolute amount

Organisation Name	errors	issued	% errors
nazwa.pl sp. z o.o.	1.138	271.919	0,41
AC Camerfirma S.A.	479	1.630	29,38
Unizeto Technologies S.A.	53	57.203	0,09
home.pl S.A.	19	54.282	0,03
Dreamcommerce S.A.	5	13.222	0,03
...	...	...	..

5.7. zLint Results

Of all valid leaf certficates, only 2.283 triggered an error, and 353.292 triggered a warning. In Figure 6, GoDaddy and Sectigo triggered warnings, even though they are known to use zLint in some fashion for pre-issuance linting. This may either be due to old certificates that are still valid, or that they choose to ignore these specific lints. In Table 7, most organisations triggered only few errors relative to their total issued certificates. With almost a rate of 30% of their issued valid leaves triggering errors, AC Camerfirma S.A. is clearly an exception in our dataset. The mozilla wiki details 26 potential issues with certificates from this CA [22], and the Chromium open-source project to plans to block this CA in a future release entirely [23].

5.8. Certificate trust chains

We examined the trust chains leading from certificates to a trusted root certificate. As mentioned earlier, we didn't do any revocation checking. Therefore, we also found trust chains containing revoked intermediates. For example, the Let's Encrypt's R3 intermediate certificate signed by DST CA X3 has a twin with different content. This twin intermediate, however, according to crt.sh as of 08/01/2021 has been revoked via OCSP, and CRL by the CA. Those circumstances make it hard to interpret the gathered data, but would provide an interesting starting point for further analysis. If we describe a set of trust chains that lead to a certificate as a trust chain configuration, we can still examine the most used trust chain configurations. The top 20 distinct trust chain configurations combined lead to 98,22% of valid leaf certificates. None of the top 20 trust chain configurations contained a root certificate that directly signed a valid leaf certificate.

6. Conclusion and future work

6.1. Conclusion

We have examined several aspects of TLS certificates, and summarized them. It was possible to build a trust chain leading to a trusted root for a majority of certificates in the dataset. Let's Encrypt currently dominates the TLS certificate ecosystem. Certain attributes of a certificate are characteristic for a CA - e.g. 99.55% of all valid leaf certificates with a ECDSA-SHA256 signature are issued by Cloudflare. With zLint, almost no misissued certificates could be found in our dataset. A CA that is known to issue problematic certificates could be clearly identified.

6.2. Future Work

In this Paper we do not cover every field that may be present in a TLS Certificate. The analysis could easily be extended to cover additional fields. Also, zLint is an established certificate linter that is known to be used by some CAs [13]. More misissued certificates may be unconvered by creating additional lints that have not been published before.

References

[1] Google, "Google Blog," https://blog.google/products/chrome/milestone-chrome-security-marking-http-not-secure/, July 2018, [Online; accessed 7-January-2021].

[2] Mozilla, "Mixed Content Blocking in Firefox," https://support.mozilla.org/en-US/kb/mixed-content-blocking-firefox#w_what-is-mixed-content-and-what-are-the-risks, [Online; accessed 7-January-2021].

[3] P. R. Donahue, "Https or bust: Chrome's plan to label sites as "not secure"," https://blog.cloudflare.com/https-or-bust-chromes-plan-to-label-sites-as-not-secure/, [Online; accessed 7-January-2021].

[4] Mozilla, "How to troubleshoot security error codes on secure websites," https://support.mozilla.org/en-US/kb/error-codes-secure-websites, [Online; accessed 7-January-2021].

[5] Google, "Fix connection errors," https://support.google.com/chrome/answer/6098869?hl=en.

[6] IETF, "RFC8555," https://tools.ietf.org/html/rfc8555.

[7] M. B. Zakir Durumeric, James Kasten, "Analysis of the HTTPS Certificate Ecosystem." IMC '13: Proceedings of the 2013 conference on Internet measurement conference, October 2013.

[8] D. Kumar, Z. Wang, M. Hyder, J. Dickinson, G. Beck, D. Adrian, J. Mason, Z. Durumeric, J. Halderman, and M. Bailey, "Tracking certificate misissuance in the wild," 05 2018, pp. 785–798.

[9] IETF, "RFC 5280," https://tools.ietf.org/html/rfc5280.

[10] CAB Forum, "Baseline Requirements Documents (SSL/TLS Server Certificates)," https://cabforum.org/baseline-requirements-documents/.

[11] tumi8, "GoScanner," https://github.com/tumi8/goscanner.

[12] zMap, "zMap Project," https://zmap.io/.

[13] ——, "zLint," https://github.com/zmap/zlint.

[14] Mozilla, https://wiki.mozilla.org/CA/Included_Certificates.

[15] zMap, "zCrypto," https://github.com/zmap/zcrypto.

[16] Mustafa Emre Acer, Emily Stark, Adrienne Porter Felt, Sascha Fahl, Radhika Bhargava, Bhanu Dev, Matt Braithwaite, Ryan Sleevi, Parisa Tabriz, "Where the Wild Warnings Are: Root Causes of Chrome HTTPS Certificate Errors," *CCS'17*, 2017.

[17] Mozilla, "SecurityEngineering/Certificate Verification," https://wiki.mozilla.org/SecurityEngineering/Certificate_Verification, November 2019, [Online; accessed 30-December-2020].

[18] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, and C. Wilson, "An end-to-end measurement of certificate revocation in the web's pki," 10 2015, pp. 183–196.

[19] M. Goodwin, "Revoking Intermediate Certificates: Introducing OneCRL," https://blog.mozilla.org/security/2015/03/03/revoking-intermediate-certificates-introducing-onecrl/, 2015.

[20] The Chromium Projects, "Revoking Intermediate Certificates: Introducing OneCRL," https://dev.chromium.org/Home/chromium-security/crlsets, [Online; accessed 7-January-2021].

[21] P. Nohe, "Maximum TLS certificate validity now one year," https://www.globalsign.com/en/blog/maximum-ssltls-certificate-validity-now-one-year, [Online; accessed 7-January-2021].

[22] Mozilla, "Common CA Database," https://www.ccadb.org/.

[23] Ryan Sleevi, https://groups.google.com/g/mozilla.dev.security.policy/c/dSeD3dgnpzk/m/iAUwcFioAQAJ.

doi: 10.2313/NET-2021-05-1_10