Do CAA, CT, and DANE Interlink in Certificate Deployments? A Web PKI Measurement Study

Pouyan Fotouhi Tehrani*, Raphael Hiesgen†, Teresa Lübeck†, Thomas C. Schmidt† and Matthias Wählisch*

*TU Dresden, Dresden, Germany
Email: {pouyan.tehrani,m.waehlisch}@tu-dresden.de
†HAW Hamburg, Hamburg, Germany
Email: {raphael.hiesgen,teresa.luebeck,t.schmidt}@haw-hamburg.de

Abstract—Integrity and trust on the web build on X.509 certificates. Misuse or misissuance of these certificates threaten the Web PKI security model, which led to the development of several guarding techniques. In this paper, we study the DNS/DNSSEC records CAA and TLSA as well as CT logs from the perspective of the certificates in use. Our measurements comprise 4 million popular domains, for which we explore the existence and consistency of the different extensions. Our findings indicate that CAA is almost exclusively deployed in the absence of DNSSEC, while DNSSEC protected service names tend to not use the DNS for guarding certificates. Even though mainly deployed in a formally correct way, CAA CA-strings tend to not selectively separate CAs, and numerous domains hold certificates beyond the CAA semantic. TLSA records are repeatedly poorly maintained and occasionally occur without DNSSEC.

Index Terms—DNS, DNSSEC, CAA, PKI, TLS, CT Logs

I. INTRODUCTION

Secure and authenticated transport is essential to the modern web. Trust in the Web PKI is built on X.509 certificates and derives from accepted root certification authorities (CA). Any CA, which is part of a valid trust chain, can issue a trustworthy certificate for any service. Millions of devices rely on this ecosystem for browsing, shopping, online banking, *etc.* Given the immense reach of each CA, much effort has gone into securing the Web PKI (Figure 1)—most prominently DNS-based Authentication of Named Entities (DANE) [1], Certification Authority Authorization (CAA) [2], and Certificate Transparency (CT) logs [3].

DANE and CAA build on the Domain Name System (DNS), which allows domain owners to store accessible information. *DNSSEC* adds an intrinsic chain of trust along the DNS hierarchy. Different from the certificate chain-of-trust, only the entity that controls a domain can sign its records. Using DANE TLSA records, the DNS can provide additional information to verify X.509 certificates and even establish trust without relying on a CA. In contrast, *CAA* records allow domain owners to restrict which CAs are allowed to issue certificates for their domains. Independent of the DNS, *CT logs* are append-only databases that collect published certificate and make misissued certificates visible to the public.

All three standards—DANE, CAA, and CT logs—concern X.509 service certificates but use different methods, different publication channels, and are targeted at different audiences. In
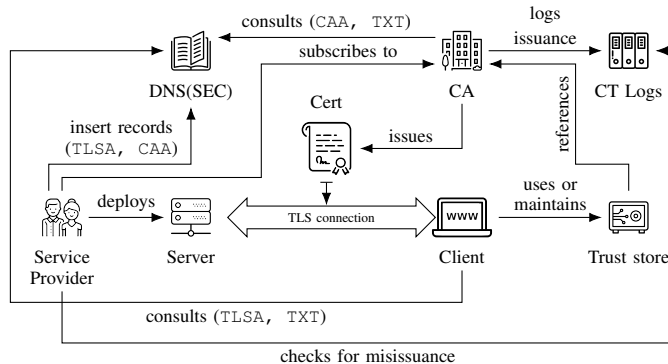
Fig. 1: Overview of Web PKI entities and their relation to DNS(SEC).

this work, we want to learn about the use (and misuse) of these concurrent approaches to harden the Web PKI ecosystem. In a large measurement study, we collect data from 4M domains based on the Tranco top list, spanning DNS records, X.509 certificates, and CT log entries. We compare and analyze the different records in use with respect to their existence, content, and intended semantics, *i.e.,* correctness, consistency, and coherence. Our major findings read:

1) Nearly 9% (357k) of all domains deploy at least one of either DNSSEC, DANE, or CAA; overlapping deployment is much smaller.
2) CAA records are largely deployed correctly and consistent with certificate issuers ($> 90\%$). Nevertheless, CA strings are not precisely defined; several strings match more than a single CA, one matches 21 CAs.
3) CAA records are mainly deployed without DNSSEC; even some TLSA records lack DNSSEC protection.
4) TLSA records are rare and often poorly maintained, some of which correspond to certificates that have been revoked or have expired since long.

The remainder of this paper is structured as follows. §II introduces the Web PKI along with the relevant technologies for securing certificate deployments and related work. Our measurement method, data collection and its processing are explained in §III. §IV reports on the deployment of X.509 certificates and the DNS extension records and examine their

correctness with respect to the actual certificates in use. §V combines information from DNS records, certificates, and logs to systematically explore consistency and coherence of the security information set. We conclude in §VII with an outlook on future research.

II. BACKGROUND AND RELATED WORK

The Web PKI ecosystem is the foundation for authentication on the Web using X.509 certificates at its core, *e.g.,* [6]. Since its inception, various extensions have been introduced to enhance its functionality or address its shortcomings. In this section, we briefly introduce Web PKI and discuss how DNS CAA, DANE, and CT Logs address one of the biggest challenges: unconstrained and global certification authority. Furthermore, we present prior work.

A. Background

A fundamental security challenge on the Internet is the trust in cryptographic keys used for authentication. In a controlled environment, keys can be manually attributed to specific entities, but such an approach is less applicable in large-scale distributed systems such as the Internet. We now discuss how Web PKI addresses this challenge, which shortcomings still exist, and which remedies have been proposed.

Web PKI. Web PKI introduces *Certification Authorities* (CA) to bind public keys to domain names (among other attributes) to form a certificate. The authenticity of a certificate can be verified using its cryptographic signature. In public key cryptography, a signature is generated by a private key (which is kept secret) and can be validated by the corresponding public key (which is published openly). A relying party (RP), *i.e.,* a piece of software that decides whether a certificate is valid or not, would then trust a certificate that is signed by or can be traced back to a trusted CA. On the Web, an RP is typically a browser, which maintains its own set of trusted CAs or Trust Anchors (TA) in a local *trust store*.

Issuing a certificate correctly is the most important task of a CA. Unfortunately, CAs are not restricted when issuing certificates. They can create certificates for any name. A compromised or malicious CA, then poses security risks for all entities that rely on security assurances provided by the Web PKI. Those incidents occur in practice, such as the DigiNotar incident [7]. To counter this threat, various solutions have been introduced, which we summarize in Table I and discuss in the following.

Certification Authority Authorization (CAA). A CAA record [2] gives a domain name owner the ability to restrict issuance of certificates by defining which CAs are allowed to issue certificates for its name. Such a *constraint* is stored in the DNS using dedicated CAA resource records to describe restrictions for wildcard or fully qualified domain names (FQDN) for the namespace under control of the name owner.

A CAA record is composed of a flag, a tag, and a value. Issuance constraints are defined by `issue` and `issuewild` tags. The latter only constrains wildcard certificates while `issue` records concern both, wildcards and FQDNs, but are superseded by an `issuewild` record. These two tags have a well-defined syntax. When the syntax is violated, a certificate should not be issued. To forbid issuance explicitly, an empty value (`";"`) can be used.

CAA records allow learning details about the CA itself. DigiCert, for example, accepts `digicert.com` as well as `amazon.com` (among others) according to its Certification Practices Statement (CPS) [8].

CAA also enables CAs to report policy violations to the name owner based on information configured in the `iodef` tag. The value of this tag can be an email address or a URL. An example of a policy violation is when a certification request is submitted at a CA that does not satisfy the issuance constraints.

DNS-Based Authentication of Named Entities. DANE [1] allows the attestation that a public key is a valid key for a domain name. To enable this attestation, a domain name owner stores the public key in specific records of the name under attestation. For TLS-based services, DNS TLSA records signal RPs which (type) of certificate to expect from the server. This can be an end entity (EE) certificate (*e.g.,* a leaf certificate) or a TA certificate (*e.g.,* from a CA). A TLSA record can reference a certificate or its subject public key information (SPKI). The reference is either to the full raw data (*e.g.,* hex formatted certificate) or its digest (*e.g.,* SHA256 hash).

DANE makes use of DNS Security Extensions (DNSSEC) which bring authentication and integrity assurance to the DNS [9] and mitigate common DNS attacks such as cache poisoning. DNS records are signed in DNSSEC, have limited validity, and must be signed again after expiration. Without DNSSEC, DNS records are susceptible to spoofing and manipulation, thus defeating the purpose of DANE.

Certificate Transparency Logs. CT logs [3], [4] were introduced to enable public monitoring and auditing of issued certificates. Although logging is not mandatory [10], major browsers such as Chrome and Safari only accept certificates that are logged in at least two compliant CT logs. Due to their market shares, this forces CAs to comply or lose out on large shares of customers.

Target Audiences. Each technology follows a different core idea and is targeted at different parties, see Table I. Whereas CAA records are meant for CAs *before* a certificate is issued, CAs feed CT logs *after* issuance to make certificates visible to domain owners and thus allow verification. DANE is deployed by domain owners as well, but faced towards certificate consumers, *i.e.,* relying parties, signaling which (type of) certificate to expect or accept.

B. Related Work

Efforts to secure the Web PKI have been ongoing for more than 10 years. CAA records were standardized in 2013 [2] and updated in 2019 [5]. DANE became a standard in 2012 [1], and CT logs were standardized in 2013 [3] and updated to version 2 in 2021 [4]. Since 2017 the CA/B forum requests that CAs validate CAA records. To the best of our knowledge,

Technology	Core Idea	Infrastructure	Responsible	DNSSEC	Target Audience
CT Logs [3], [4]	Auditable certificate issuance	CT Logs	CA	n/a	Subscribers (Domain Owners)
DANE [1]	Bind public keys to names	DNS	Domain Owner	mandatory	Relying Party (Clients)
CAA [5]	Constraint issuer of domain names	DNS	Domain Owner	not mandatory	CA (Certificate Issuer)

TABLE I: CT Logs, DANE, and CAA offer three different approaches to securing the Web PKI.

this paper is the first study that comprehensively analyzes all these technologies together to better understand configurations in real deployments, including inconsistencies.

CAA. In 2018, Scheitle *et al.* [11] presented the first analysis of the CAA ecosystem, actively measuring the behavior of selected CAAs as well as auditing the ecosystem. They found that 3k of 95k domains in the Alex Top 1M list deployed CAA records. At that time, most domain owners (89%) configured a single CAA string, mainly letsencrypt.org (64%), and did not allow issuing certificates for arbitrary names of a domain or subdomain (59%). CA strings that occurred infrequent included many invalid strings due to misspellings or owners using their own domain as CA string—indicating a lack of automation and understanding of how to configure CAA. In a study focusing on nonfederal governments in the United States, Gebhard *et al.* [12] revealed that CAA is least deployed and grows slowly compared to the adoption of DNS records that strengthen identifying web servers (SPF, DMARC).

Certificate Transparency. In 2018, the same year Chrome made CT mandatory, Scheitle *et al.* [13] measured the adoption of CT and found an exponential increase of CT log entries. They further found two more use cases of CT logs. (*i*) CT logs can be used to identify phishing domains, (*ii*) malicious actors monitor CT logs to search for new targets. The latter was examined in a broader study in 2021 [14], which confirmed the continued use of CT logs for target discovery. In a longitudinal study of TLS certificates gathered from active scanning and CT logs, Farhan *et al.* [15] found an improvement in share of valid certificates and key strengths, but also observe a centralization in the Web PKI. 80% of valid certificates are now signed by only 10 keys.

DANE. Three years after DANE was published, Zhu *et al.* [16] found in 2015 that less than 1000 domains use DANE. In 2020, DANE still did not gain widespread adoption in browsers. While the email ecosystem saw a comparatively higher adoption rate, mismatches between TLSA records and certificates and incorrect DNSSEC were still frequent [17]. Lee *et al.* [18] observe that 94% of SMTP servers still rely on the certificates issued by CAs when they deploy DANE.

All the prior studies provide an in-depth understanding of CAA, CT, or DANE deployments. They focus, however, on each protection mechanism separately. In this paper, we first provide an update of recent deployments and then close a gap by taking a comprehensive view on the deployments of all three technologies together.

III. METHOD AND DATA CORPUS

We collect a data corpus of 4M domains to examine the Web PKI ecosystem[1] using the processing pipeline in Figure 2.

A. Building a Target List

Query DNS (T.1). As input, we take the Tranco top list [19] comprised of over 4.1M domain names ranked by popularity. We try to resolve each name to an A record and only keep names with valid records. $\approx 533k$ entries did not resolve to an IP address while 1150 timed out, likely due to trimmed domain names or dynamic DNS changes.

Check Ports (T.2) & Transport (T.3). Next, we check ports 80 and 443 (TCP using `netcat`) and establish an HTTP or HTTPS connection over open ports with our own tool based on `libcurl`. We follow HTTP (`3xx`) and HTML (`http-equiv`) redirects. $145k$ domains allow neither HTTP nor HTTPS connections. The remaining names resolve to IP address and host a web server.

Browser (T.4). We then feed collected names into Puppeteer (T.4), a headless Chrome browser, and follow further redirects (including JavaScript). Our target list now contains $4M$ unique domain names (including intermediates).

B. Collecting DNS Records & Certificates

CAA, DANE, and DNSSEC from DNS (M.1). We collect the following DNS resource records (RR) for each domain: SOA, A, CAA, TLSA, as well as *contactemail* and *contactphone* TXT records, see Appendix C. Additionally, we query CAA records for all parents of a given name by recursively removing the leftmost label up to the TLD, *e.g.,* for `www.example.co.uk` we query CAA records for the domain set {`www.example.co.uk`, `example.com`, `co.uk`}. Queries set the `DO` flag to request DNSSEC records (if any) and have DNSSEC validated by the resolver. For DANE, we only query TLSA records that are associated with TCP services on port 443 (HTTPS).

We use Google recursive resolvers for all DNS queries due to their availability, reliability, and provision of a JSON API.

X.509 Certificates via TLS (M.2). For each domain name, we establish a TLS connection with the first retrieved IP address in our list, setting the domain name as the Server Name Indication (SNI). At this point, we do not validate certificates and we suppress TLS errors due to insecure cipher suites (OpenSSL security *Level 0*). By disregarding failures in TLS
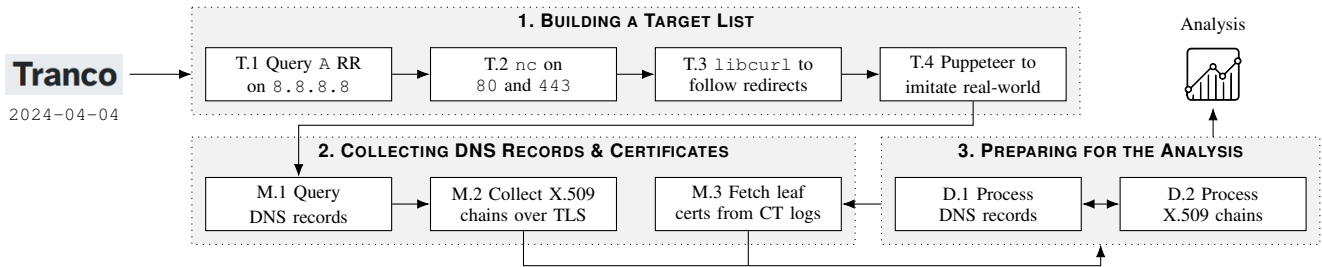
Fig. 2: Simplified illustration of our toolchain for building a target list, collecting a dataset, and preparing data for analysis.

deployments, we can collect all available certificates to better understand shortcomings.

Leaf Certs from CT Logs (M.3). Domain owners might have applied for more certificates than the one deployed on their web server. For example, Cloudflare issues *backup certificates* for its customers to be able to swiftly replace keys in case of compromise [20]. We collect certificates appended to CT logs for the subset of domain names that have either CAA or TLSA records, using the open database provided by Sectigo under `crt.sh`.

C. Preparing for the Analysis

Parsing and Validating CAA (D.1). We parse CAA `issue` and `issuewild` values according to the Augmented Backus-Naur Form (ABNF) and consider malformed entries as empty (semantically equivalent to `";"`). Malformed entries forbid the issuance of a certificate [5]. For records with `iodef` tags, we verify that the value is a URL with the correct scheme (`mailto`, `http`, or `https`).

For records with the `issue` and `issuewild` tags, we devise an algorithm that matches a set of given CAA RRs to certification authorities, visualized in Appendix D. It uses the following classifications:

1) **No CAA**: No applicable CAA RR was found.
2) **Implicit Match**: No CAA RR constraints issuance.
3) **Issuer Match**: At least one CAA RR matches cert issuer.
4) **Issuer Mismatch**: No CAA RR matches the issuer.
5) **Malf. Mismatch**: All CAA RR are malformed.
6) **Empty Mismatch**: Only empty (`";"`) CAA RR.

This algorithm relies on a mapping from CAA issuer domain names (*i.e.,* values in `issue` and `issuewild`) to CA certificate properties. Our mapping is based on the "List of CAA Identifiers" in the Common CA Database [21]. We enrich this list manually (*i*) based on in Certification Practice Statements (CPS) documented identifiers, and (*ii*) our own observations of undocumented string identifiers. The CPS is usually linked in the certificate. If this link is not valid, we manually identify the respective CA and browse its website to find the statement.

In contrast to X.509 certificates or DNSSEC signatures, CAA RRs do not carry a validity timestamp and are in practice only validated by the CA at the time of issuance. This can cause a discrepancy between our observations at the time of measurement and what CAs observed when they issued a certificate, and thus leads to misclassifications. To verify whether
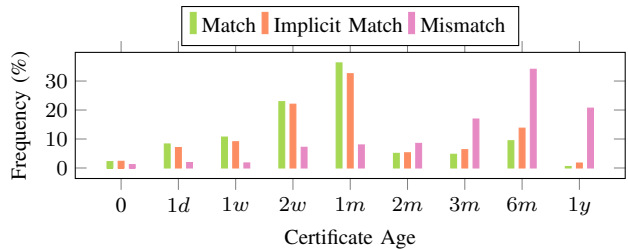

Fig. 3: CAA matching states by age of certificates with matching subject name at the time of measurement. 75% of certificates are younger than 3 months. CAA mismatches appear relatively more often in certificates older than 3 months.

our measurement setup introduces such misclassifications, we calculate the difference between our probe time and `not before` timestamp carried in certificates for domains with CAA records, see Figure 3. Regardless of CAA matching state, our measurements occurred in 75% within three months of the issuance. Mismatches, however, occur much more frequently for certificates that are older. This indicates that our setup is not prone to mismatches that are actually valid.

Parsing and Validating DANE (D.1). We validate that TLSA records are RFC-conformant using an open-source library [22] but leave the verification of the DNSSEC integrity to the recursive resolver.

Parsing and Validating X.509 Certificates (D.2). We parse X.509 certificates with ZCrypto [23] and check three things. (*i*) The subject (alternative) names in the certificate should match the queried domain name, *i.e.,* the name is included as a SAN or covered by a wildcard SAN. (*ii*) The certificate chain should be valid. We select the Mozilla Common CA Database [21] as our trust store. And (*iii*) Attached SCTs (if any) should be valid, *i.e.,* the SCT corresponds to the respective certificate and is signed by a trustworthy log. We use Google's library [24] to verify signatures with keys extracted from the list of all complying [25] logs [26]. Here, we assume that log operators behave correctly and omit to check the logs directly.

IV. CONFIGURATION OF X.509 CERTIFICATES, DNS CAA, AND DANE RECORDS

In this section, we focus on the deployment of X.509 certificates alongside CAA and DANE records within the DNS to better understand to which extent name owners care about
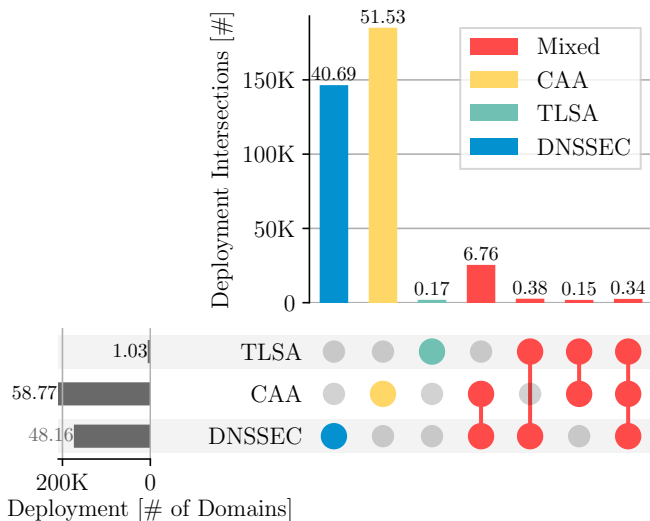
Fig. 4: The UpSet plot contrasts the use of DNSSEC, CAA, and TLSA among domains that deploy at least one of them. CAA is most deployed ($\approx$52%) followed by DNSSEC ($\approx$41%) but they only co-occur in $<$7% of domains. DANE (TLSA) is rarely deployed ($\approx$1%).

correct configuration of each security extension. In total, we observe 357k unique domains (8.85% of all names in the Tranco list) that deploy at least CAA, DANE, or DNSSEC. We visualize the overlap of support for different technologies per name as an UpSet plot [27] in Figure 4. The bar plot on the left shows how many domains fall into each category while the bar plot on top shows the size of exclusive intersections between the categories marked in the matrix below. It is clearly visible that domain name owners do not focus on comprehensive security support.

A. X.509 Certificates

Roughly 97% of collected certificates are valid (based on our trust store, see §III-C). Validation failures stem from age ($47k$), untrusted signees ($\approx 15k$), and malformed certificates ($60k$). All valid certificates were submitted to either 2 (72%), 3 (23%), or 4 (5%) CT logs. With 73%, most certs were submitted to logs operated by Google, two thirds were submitted to logs operated by Cloudflare, and about half to logs operated by DigiCert.

Two out of every three certificates were issued by one of four major CAs: Let's Encrypt with a total share of 52%, followed by Google Trust services (16%), DigiCert (4.5%), and Sectigo (4.5%). This does not include resellers, *e.g.,* ZeroSSL using Sectigo infrastructure.

In our observation, 7.2% of unique domains point to hosts that provide certificates with mismatching subjects, *i.e.,* not matching the original domain name. These are in part default (self-signed) server certificates or service provider certificates for parked domains.

CAA String	issue		issuewild		CA Count[†]
	Overall	Single	Overall	Single	
letsencrypt.org	86.95%	17.70%	85.28%	6.43%	1
digicert.com	62.58%	1.31%	77.94%	1.36%	4
comodoca.com	45.68%	0.39%	73.73%	0.34%	12
pki.goog	33.27%	0.91%	51.10%	0.06%	1
globalsign.com	32.01%	0.55%	31.13%	0.63%	2
sectigo.com	21.52%	1.04%	25.37%	1.79%	21

[†] Count of unique CAs (by Subject Organization) in our dataset that match the CAA string in the first column

TABLE II: The CA strings that appear in more than 10% of the CAA records in our dataset as well as their relative occurrence as the *only* CA string.

B. CAA Deployment

5.2% of the $4M$ domains we scanned support CAA records. Most (4.55%) domains only specify constraints for CAs for fully qualified domain name or wildcard names (respectively issue or issuewild records). Around 0.01% only provide information on reporting policy violations (iodef records). 0.63% of domains deploy both.

Reporting Policy Violations—iodef Records. Several records that have been configured prevent contact because of misconfiguration. About 3.79% of iodef entries are invalid due to an invalid schema [5] as neither mailto nor http[s] is present. 17 entries with unknown schemas still contain a colon, eight of these use a non-existent schema (mailinto:, mail:, etc.), six are typos with missing or switched letters, and three contain characters that break the formatting. Among the remaining invalid entries, 924 are likely email addresses and 58 are HTTP endpoints. One record only contains a sequence of 27 numbers, which is unlikely a phone number because it is too long. Among valid records, nearly all iodef records contain email addresses ($> 99\%$), 42 in total an HTTPS URL, and 16 domains have both.

We contacted domain owners with invalid iodef records. Since many email addresses were associated with multiple domains, the 924 entries could be reduced to 504 distinct addresses. Nearly 16% of the mails could not be delivered, mostly because the mailboxes no longer exist. While most did not respond, we also got kind responses and learned that at least one case was the result of false documentation of a hosting provider. Others mentioned that their DNS settings were applied automatically by the service provider.

Granting Authorization to Issue (Wildcard) Domain Names—issue and issuewild Records. Among $\approx$210k domains with issuance constraints, 97% have records with an issue tag and 57% have records with an issuewild tag. The overlap is 54%, which leaves 43% that only have issue and 3% that only have issuewild. 38 domains have malformed entries.

Our dataset shows that compared to six years ago, name owners are more liberal when it comes to allowing multiple CAs to issue certificates. In 2018, 89% of domains only allowed a single CA to issue certificates [11]. Now, in 2024,
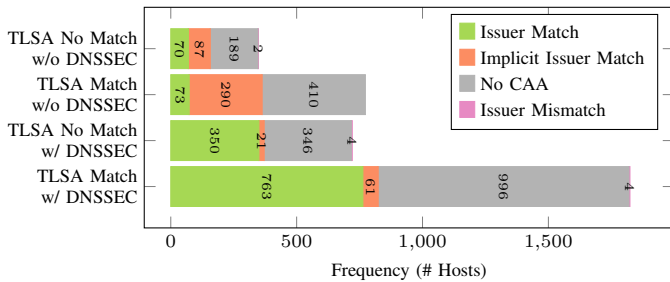
Fig. 5: Total number of hosts delivering valid certificates with valid TLSA and CAA records divided by DNSSEC support, and TLSA and (selected) CAA matching status.

Fig. 6: Most certificates with CAA deployment are consistent with their CAA records. We only observed an "Issuer Mismatch" in 2237 domains (1.07% of CAA records).

28.39% domain owners allow four different CAs to certificate their names, 16.35% 3 CAs, 16.26% 5 CAs, and 10.04% 2 CAs. We even observe one domain with 59 CAA `issue` records and another one with 46 records. Another 59 domains, all ending in `.ba.gov.br`, each have 17 CAA records.

Table II shows the most commonly used CAA strings as well as the number of unique CAs (by Subject Organization) that match this string in our dataset. Clearly, a domain owner grants permission to the operator of the CA infrastructure rather than to a specific CA. For example, ZeroSSL, an Austrian CA, accepts `sectigo.com`, because it uses Sectigo Infrastructure to issue certificates under its own brand. At the same time, a CA might accept different CAA strings. In our example, ZeroSSL also accepts `usertrust.com`, another brand of Sectigo.

Records with Non-Standard Tags. We find 353 CAA records with tags not defined in RFC [5]. These tags can be categorized in three types: (*i*) 252 × unrecognized tags as defined in CA/B Baseline Requirements [10, A.1.1.], (*ii*) 50× misspellings such as an extra letter, and (*iii*) 51× malformed formats such as extra quotes.

C. DANE Deployments

About 0.1% of all unique names that we queried are standard compliant. 98% of these 3678 names provide a valid certificate. The majority (87%) define only end-entity constraints (DANE or PKIX EE), 10% only trust anchor constraints, and the rest both. Although DNSSEC is a requirement for DANE, one out of every third TLSA record set is delivered over 1126 insecure DNS sets. Figure 5 summarizes our findings.

It is noteworthy that only 70% of names match their TLSA records as described below. The case of mismatching TLSA records is discussed in depth in §V using data from CT logs.

Matching TLSA with Invalid Certificates. About 1.5% (53 names) define matching DANE-TA or DANE-EE constraints with invalid certificates (see §III). More than half are expired and issued by Let's Encrypt and the remaining rest are self-signed certificates or leaf certificates by miscellaneous (partly not accredited) CAs. More than 71% of TLSA record sets here are secured by DNSSEC and 73% have no associated CAA records.
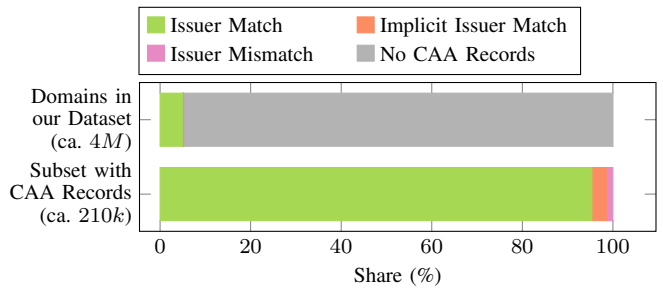
Matching TLSA with Valid Certificates. 2553 entries provide TLSA records that match the provided certificate and the certificate is valid. However, not all are secured by DNSSEC and only about 46% of entries provides CAA records. Only 190 names in this set have TLSA records that impose a limit on Web PKI certificates, while the rest are DANE-EE and DANE-TA constraints.

V. INFORMATION CONSISTENCY

A. CAA Records vs X.509 Certificates

Based on the results of our CAA matching algorithm, described in §III-C, we examine the consistency between CAA records and the certificate issuers of the TLS certificates. While the RFC [5] mentions the role of an auditor ("Certificate Evaluators"), CAA records provide information for CAs at the time of certificate issuance and are not required to stay consistent.

CAA Validation Overview. Figure 6 shows the result of our classification for two datasets based on CAA records with the `issue` tag. The upper bar contains all 4M domains in our dataset while the lower bar only considers the subset with CAA records. 3.8M domains (94.8%) do not have CAA records.

200k domains (4.96%) have certificates consistent with their CAA records ("Issuer Match"). These are 95.34% of the nearly 210k domains with a CAA record. Another 0.17% (3.22%) is classified as "Implicit Issuer Match". These domains mostly (93%) have CAA records with the `issuewild` tag but not with an `issue` tag while containing FQDN in their certificate, *i.e.,* the domain has the relevant resource records but only restricts the issuance for wildcard certificates. 6% only deploy `iodef` records. < 1% only have records with unknown tags.

Nearly 91% of domains with CAA records deploy the relevant RR, *i.e.,* the CAA record(s), themselves. For roughly 9% the direct parent domain has the relevant RR. For about 200 domains (≤1%) the DNS hierarchy needs to be traversed further, up to 4 times, which occurred only once.

Only 0.06% of all domains (1.07% with CAA records) have an "Issuer Mismatch", *i.e.,* the string in their CAA records

does not match the issuer of their certificate. We found two domains that deployed *only* malformed CAA records.

Relevant CAA Records. For most domains (92.4%) a CAA record with the `issue` tag was the deciding record, *i.e.,* the CAA record that fit the domain we requested from the web server. In 95% of cases the issuer matches, 3.5% have an implicit match, and the remaining 1.5% mismatch. In cases where the CAA record with the `issuewild` tag was relevant (7.6%), the share of domains with a matching issuer is even higher with 99.4%. 0.6% mismatch, and we do not observe any implicit matches.

Wildcard Certificates for Subdomains. Next, we examine domains with `issuewild` CAA records and a wildcard name for their *subdomains* in the certificate. This is the case for 21% of domains with CAA records in our dataset ($44k$). For 99.5% domains the issuer of the certificate is consistent with their CAA `issuewild` records. 216 (0.49%) have a mismatch, *i.e.,* their certificates should not have been issued in this configuration. 17 (0.04%) forbid issuance of a wildcard certificate with an empty CA string (but have an active wildcard certificate).

Partial CAA Matches. A CAA `issue` record covers a FQDN and a wildcard domain in the absence of an CAA `issuewild` record. However, a CAA `issuewild` record without an `issue` record would only restrict wildcard issuance. If a certificate lists both FQDN and a wildcard domain, CAs should check for CAA `issue` and `issuewild` records. A *partial* match occurs when the issuer of a certificate with a FQDN and a wildcard matches either an `issue` or `issuewild` record, but not both.

For 60 domains, we can match the certificate issuer to the CAA `issue` record but observe a mismatch for the wildcard in the same certificate. And for the reverse—the certificate issuer matches a CAA `issuewild` record, but not the `issue` record—we find 151 occurrences. In no case is issuance restricted by an empty (';') record.

It seems unlikely that domains, which change their CAA records in the time between re-issuance, change only parts of their records and that they do not choose to set an empty (';') record. As such, these are likely falsely issued certificates.

B. CAA Records vs CT Logs

In §V-A we match the issuer of the certificates from web servers against their respective CAA records. For all domain names with at least one CAA record, we query CT logs to fetch all other certificates bound to those names that are valid at the time of measurement. CT logs provide $1M$ certificates for about $191k$ unique domain names—reduced to $766k$ certificates after removing duplicates (*e.g.,* when a CA logs both precert and leaf). To compare the results, we focus on domains with CAA records or mismatching server certificates.

In 98% of cases the CAA matching state is consistent, *i.e.,* the certificates we retrieved from the web server and the certificates we retrieved from the CT logs have the same consistency with the domain's CAA records. Among inconsistent
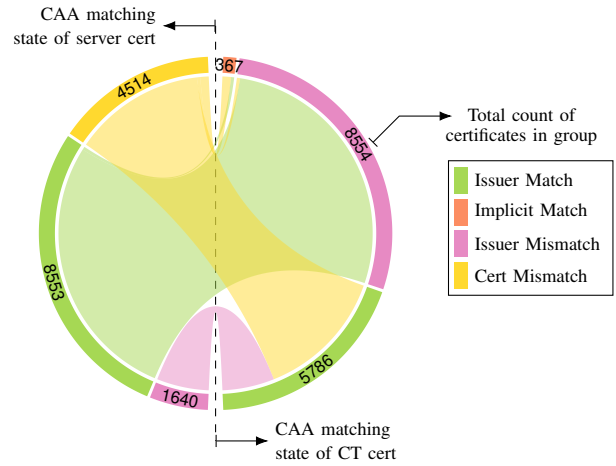

Fig. 7: Distribution of inconsistencies between CAA matching state of server (left half-circle) and CT-logged certificates (right half-circle).
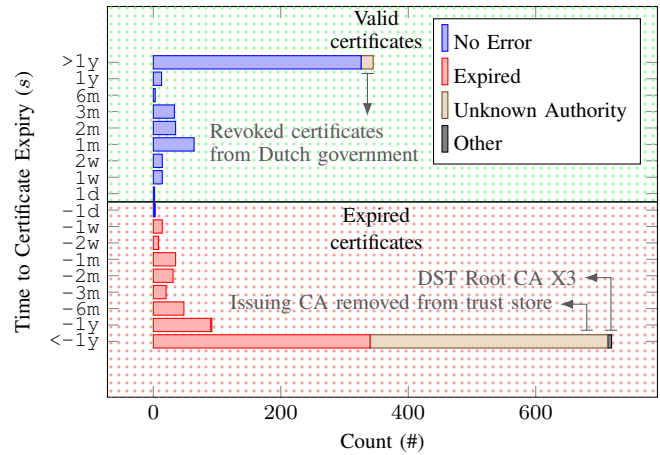

Fig. 8: Certificates from CT logs that match TLSA records but are not deployed on the web server, grouped by their relative expiration time and annotated with their validation state.

cases, $\approx 55\%$ of domains have servers that return a CAA-matching certificate while CT logs contain at least one other valid certificate that does not match the CAA constraints. 30% present a certificate that does not match the domain name (see §IV) while we find a logged certificate that fulfills the CAA constrains. Figure 7 summarizes the findings.

C. TLSA Records vs CT Logs

DANE sets constraints on certificates provided by a service endpoint. We take advantage of CT logs to find certificates that are not deployed, but match either TLSA records or the domain name of DANE-enabled services. We explore (*i*) the reasons for the existence of TLSA records that do not match the server certificate, (*ii*) the relationship between valid but not-deployed certificates and TLSA records, and (*iii*) compatibility of CAA records with certs referred to by DANE.

TLSA Records that do not Match the Server Certificate.
We observe 1171 names with TLSA records that do not match
the provided certificate. Thus, DANE-enabled clients would
consider these certificates invalid. We query the `crt.sh` CT
database for certificates that match these records and find
1494 certificates (note that a single domain can have multiple
TLSA records). Figure 8 depicts how these are distributed with
respect to their validity (§III-C) and relative expiration time.

The majority of mismatching TLSA records references
certificates (leaf or CA) that are expired or have been removed
from current trust stores. An indication that TLSA records
have not been updated to reflect changes of the CA or leaf
certificate. We also see TLSA records reference existing and
valid certificates which are not deployed.

Undeployed Certs that Match TLSA Records. For 945
domain names with TLSA records that fit their respective web
server certificate we find 1002 more matching certificates in
CT logs that are *not* included in the certificate chain of the
respective web servers. The majority (784) are leaf certificates,
and the rest (218) are from intermediate or root CAs.

All leaf certificates are either renewed or expired versions
(same public key) of the certificate provided by the server.

Certificates Matching DANE-secured Domains. For all
domain names with TLSA records, we query CT logs for
matching certificates (subject or SAN) and keep only valid
ones (relative to the measurement time). We only fetch leaf
certificates and not the complete chain due to limitations in
`crt.sh` database API. Thus, we can only authenticate TLSA
end-entity (PKIX-EE or DANE-EE) records against logged
certificates. A total of 10358 certificates match 3320 unique
domain names.

Table III summarizes for how many certificates can be
authenticated by the TLSA records. Notable are cases where
a TLSA record matches the server cert but not the CT
cert and vice versa (rows #3 to 6). The majority of server-
match/CT-mismatches (row #3) are TLSA records that specify
a certificate by its fingerprint (and not SPKI), so that even if
the same key is renewed, it will not match. We also observe
Cloudflare backup certificates (of row #4), where multiple
certificates are valid simultaneously at the same time.

D. TLSA Records vs CAA Records

Most DANE-secured certificates collected from CT logs
have the same CAA matching status as their corresponding
server certificates. The only notable exception is a set of
domain names with matching CAA records and TLSA records
referencing a valid certificate that does not match CAA (19
in total). We also observe 19 cases where the server returned
a certificate that does not match its domain name, but TLSA
records reference certificates that match CAA constraints (9
already expired).

TABLE III: Number of CT logged certificates for all DANE-
secured domain names grouped by authentication status.

| # | TLSA Record Authenticates: | | Same Issuer (Server and CT) | Count |
	Server Cert	CT Cert		
1	✓	✓	✓	959
2	✓	✓	✗	1
3	✓	✗	✓	2,432
4	✓	✗	✗	294
5	✗	✓	✓	70
6	✗	✓	✗	15
7	✗	✗	✓	1,777
8	✗	✗	✗	431

VI. DISCUSSION

In this section, we discuss our findings comprehensively and
provide recommendations for name owners and CA operators.

Lack of DNSSEC deployment defeats CAA purpose.
Missing DNSSEC deployment can be abused to trick CAs
into misissuing domain-validated certificates [28]. While the
CAA standard does not require CAA records to be secured
by DNSSEC, the use of DNSSEC is "strongly RECOM-
MENDED" [5]. Yet, we observe a small overlap in the
deployment as only $\approx 7\%$ of CAA-enabled domains are also
DNSSEC-secured, opening an unnecessary attack surface.

**Reused strings in CAA *issuer-domain-name* weaken se-
curity.** Not only do some CAs accept multiple strings, but
some strings are accepted by multiple CAs. We found one
string accepted by up to 21 CAs, see Table II. While these are
likely resellers or managed PKIs, this nonetheless weakens
the restrictiveness the system intends to implement. Users are
likely unaware of this when they choose their CA and create
CAA records. This conflicts with the goal of CAA records.

**An authoritative source for CAA mappings can im-
prove deployment and security.** We found many invalid
strings clearly indicating that CAA records are configured
manually, even though supportive tooling (*e.g.,* `https://
sslmate.com/caa/`) exists. A well-defined interface to dis-
cover legitimate CAA strings of a CA could help avoid typos,
ease the setup, and increase the stability of the ecosystem.
Furthermore, the current specification and deployment model
impacts auditors who must regularly scan the Certification
Practice Statements (CPS) of CAs to manually collect all valid
CAA strings, which may lead to incorrect and incomplete data
sets.

CT Logs can help CAA evaluators. A CA only validates
issuance constraints at the time of certification. An evaluator,
however, can validate CAA only after certificate issuance and
might face CAA records other than what the CA observed.
This also explains why the number of CAA mismatches grows
when the certificate age increases. This time discrepancy can
be reduced to a minimum by also acting as a CT Log Monitor
and validate CAA records as soon as a certificate (or its

precert) is logged.

DANE can enhance CAA. The CAA standard [5] claims that DANE records are relevant *after* certificate issuance. In fact, we found that DANE records are beneficial along the whole process and can even partly substitute CAA features. A TLSA TA record referencing root or intermediate certificates of a CA, for example, can be used to convey the same semantics as a CAA `issue` tag. Furthermore, a "to be signed certificate" provided to the CA for signing already carries the public key and can be referenced by a TLSA record before certification.

Third party validation can help name owners. Our correspondences with operators (§IV) reveal that errors in CAA entries are partly due to incorrect instructions or automatic configurations by service operators (*e.g.,* hosting and DNS providers). In addition, lack of feedback (using `iodef`) by issuing CAs in face of misconfigured or malformed CAA records makes it harder for name owners do detect such errors. To address this, we have provided a public validation tool available on `https://caa.secnow.net`, which can validate CAA records, provide comprehensible explanations on effect of each CAA record, and detect errors and suggest solutions.

VII. CONCLUSION

In this paper, we analyzed the Web PKI ecosystem with a focus on consistency between CAA, DANE, CT Logs, and X.509 certificates. Ideally, each protection mechanism is correctly configured, and all agree on the set of legitimate certificates of a name. Our results show different deployment behavior. Most alarming, a small share of certificates should not have been issued since they contradict CAA records in place. Furthermore, we found several configuration mistakes that could be prevented by better tooling and explicit best common practices. We argue that there is room for improvement. The CAA record structure itself is simple. Clear information about the CAA strings and better support for configuring CAA records can help to increase adoption and help avoid common errors we identified.

We encourage domain owners to maintain their CAA records and keep them consistent with deployed certificates. Even more, we encourage extending the deployment to include DNSSEC and to consider TLSA records, as these techniques put web security on much firmer grounds. Enabling automated evaluation of issued certificates and consider data of all possible protection mechanisms would be one step towards a consistent and healthy Web PKI.

REFERENCES

[1] P. Hoffman and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA," IETF, RFC 6698, August 2012. [Online]. Available: https://doi.org/10.17487/RFC6698

[2] P. Hallam-Baker and R. Stradling, "DNS Certification Authority Authorization (CAA) Resource Record," IETF, RFC 6844, January 2013. [Online]. Available: https://doi.org/10.17487/RFC6844

[3] B. Laurie, A. Langley, and E. Kasper, "Certificate Transparency," IETF, RFC 6962, June 2013. [Online]. Available: https://doi.org/10.17487/RFC6962

[4] B. Laurie, E. Messeri, and R. Stradling, "Certificate Transparency Version 2.0," IETF, RFC 9162, December 2021. [Online]. Available: https://doi.org/10.17487/RFC9162

[5] P. Hallam-Baker, R. Stradling, and J. Hoffman-Andrews, "DNS Certification Authority Authorization (CAA) Resource Record," IETF, RFC 8659, November 2019. [Online]. Available: https://doi.org/10.17487/RFC8659

[6] P. F. Tehrani, E. Osterweil, J. Schiller, T. C. Schmidt, and M. Wählisch, "Security of Alerting Authorities in the WWW: Measuring Namespaces, DNSSEC, and Web PKI," in *30th The Web Conference (WWW'21)*. New York, USA: ACM, April 2021, pp. 2709–2720. [Online]. Available: https://doi.org/10.1145/3442381.3450033

[7] J. Amann, O. Gasser, Q. Scheitle, L. Brent, G. Carle, and R. Holz, "Mission accomplished?: HTTPS security after diginotar," in *Proc. of ACM Internet Measurement Conference (IMC)*. New York: ACM, Nov. 2017. [Online]. Available: https://doi.org/10.1145/3131365.3131401

[8] DigiCert. (2024, Jan.) Certification Practices Statement. [Online]. Available: https://www.digicert.com/content/dam/digicert/pdfs/legal/digicert-cps-v6-05.pdf

[9] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS Security Introduction and Requirements," IETF, RFC 4033, March 2005. [Online]. Available: https://doi.org/10.17487/RFC4033

[10] CA/Browser Forum, "Baseline Requirements for the Issuance and Management of Publicly-Trusted TLS Server Certificates," Online, Jan. 2024. [Online]. Available: https://cabforum.org/uploads/CA-Browser-Forum-TLS-BRs-v2.0.2.pdf

[11] Q. Scheitle, T. Chung, J. Hiller, O. Gasser, J. Naab, R. van Rijswijk-Deij, O. Hohlfeld, R. Holz, D. Choffnes, A. Mislove, and G. Carle, "A First Look at Certification Authority Authorization (CAA)," *SIGCOMM Comput. Commun. Rev.*, vol. 48, no. 2, pp. 10–23, may 2018. [Online]. Available: https://doi.org/10.1145/3213232.3213235

[12] A. Gebhard and D. Perouli, "Poster: Measuring Adoption of SPF, DMARC, and CAA DNS Records with Nonfederal Governments in the United States," in *Proc. of the IEEE ICNP*, 2023, pp. 1–2. [Online]. Available: https://doi.org/10.1109/ICNP59255.2023.10355622

[13] Q. Scheitle, O. Gasser, T. Nolte, J. Amann, L. Brent, G. Carle, R. Holz, T. C. Schmidt, and M. Wählisch, "The Rise of Certificate Transparency and Its Implications on the Internet Ecosystem," in *Proc. of ACM Internet Measurement Conference (IMC 2018)*. New York, NY, USA: ACM, October 2018, pp. 343–349. [Online]. Available: https://doi.org/10.1145/3278532.3278562

[14] S. Pletinckx, T.-D. Nguyen, T. Fiebig, C. Kruegel, and G. Vigna, "Certifiably Vulnerable: Using Certificate Transparency Logs for Target Reconnaissance," in *Proc. of the Euro S&P*, 2023, pp. 817–831. [Online]. Available: https://doi.org/10.1109/EuroSP57164.2023.00053

[15] S. M. Farhan and T. Chung, "Exploring the Evolution of TLS Certificates," in *Proc. of PAM*, ser. Lecture Notes in Computer Science, A. Brunström, M. Flores, and M. Fiore, Eds., vol. 13882. Springer, 2023, pp. 71–84. [Online]. Available: https://doi.org/10.1007/978-3-031-28486-1_4

[16] L. Zhu, D. Wessels, A. Mankin, and J. Heidemann, "Measuring DANE TLSA Deployment," in *Traffic Monitoring and Analysis*. Springer International Publishing, 2015, pp. 219–232. [Online]. Available: https://doi.org/10.1007/978-3-319-17172-2_15

[17] H. Lee, A. Gireesh, R. van Rijswijk-Deij, T. T. Kwon, and T. Chung, "A Longitudinal and Comprehensive Study of the DANE Ecosystem in Email," in *Proc. of the USENIX Security Symposium*. USENIX Association, Aug 2020, pp. 613–630. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/lee-hyeonmin

[18] H. Lee, I. Ashiq, M. Müller, R. van Rijswijk-Deji, T. Kwon, and T. Chung, "Under the Hood of DANE Mismanagement in

SMTP," in *Proc. of the USENIX Security Symposium*. Boston, MA: USENIX Association, Aug. 2022. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/lee

[19] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, "Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation," in *Proc. of the 26th Annual Network and Distributed System Security Symposium (NDSS)*, ser. NDSS 2019. The Internet Society, Feb. 2019.

[20] D. Kozlov. (2022, Mar.) Introducing: Backup Certificates. [Accessed 2023-05-23]. [Online]. Available: https://blog.cloudflare.com/introducing-backup-certificates/

[21] "Common CA Database," website. [Online]. Available: https://www.ccadb.org/resources

[22] S. Huque, "dane. Go library for DANE authentication," GitHub. [Online]. Available: https://github.com/shuque/dane

[23] "ZCrypto. Liberal Go TLS + X.509 Library for Research," GitHub. [Online]. Available: https://github.com/zmap/zcrypto

[24] "Certificate Transparency: Go Code," GitHub. [Online]. Available: https://github.com/google/certificate-transparency-go

[25] Google. (2022, Mar.) Chrome Certificate Transparency Policy. [Online]. Available: https://github.com/GoogleChrome/CertificateTransparency/blob/cc113ef3ca489a25b4aac86cdcf19ec668c58177/ct_policy.md

[26] "Certificate Transparency: Known Logs," website. [Online]. Available: https://certificate.transparency.dev/google/

[27] A. Lex, N. Gehlenborg, H. Strobelt, R. Vuillemot, and H. Pfister, "UpSet: Visualization of Intersecting Sets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1983–1992, 2014. [Online]. Available: https://doi.org/10.1109/TVCG.2014.2346248

[28] L. Schwittmann, M. Wander, and T. Weis, "Domain Impersonation is Feasible: A Study of CA Domain Validation Vulnerabilities," in *Proc. of IEEE Euro S&P*, 2019, pp. 544–559. [Online]. Available: https://doi.org/10.1109/EuroSP.2019.00046

APPENDIX A
ETHICAL CONSIDERATIONS

Our measurements use common place actions (requesting DNS records, performing TLS handshakes) and were spread out to avoid hitting the rate limits or other protective measures. The information we collect is public and explicitly designed to be checked by the relevant parties. We neither reveal new security vulnerabilities nor do we single out any entity specifically. We individually notified all parties that were negatively impacted by misconfigurations or notable errors.

APPENDIX B
X.509 CERTIFICATES

In the Web PKI X.509 build a chain of trust from a self-signed *Root* certificate over one or more *Intermediate* certificates to a *Leaf*. As shown in Figure B.1 the hierarchy is expressed through signatures from the root towards the leaf in one direction and references to the issuer in the other direction.

APPENDIX C
MEASUREMENT DETAILS

Table C.1 lists when each measurement task described in §III was performed and which tools we used. Table C.2 provides additional information about the DNS queries when collecting X.509 Certificates via TLS (M.2).

APPENDIX D
CAA MATCHING ALGORITHM

The consistency between CAA records and the certificate issuer can be classified into three main groups (colors in Figure D.1): No CAA (relevant CAA RR set is empty), Issuer
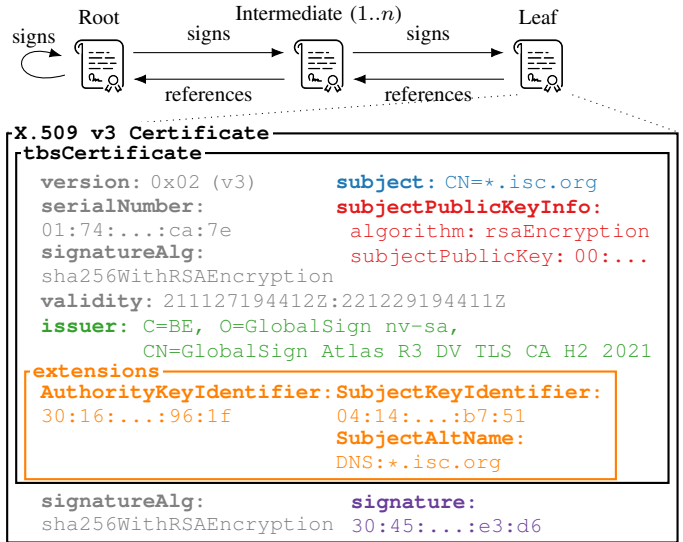

Fig. B.1: X.509 certificate chains and structure.

TABLE C.1: Measurement dates and deployed tools for each task in our pipeline visualized in Figure 2.

Task	Name	Date	Tool
T.1	Query AA Records		`dig`
T.2	Check ports 80/443	2024-04-07	`nc`
T.3	Follow HTTP/HTML Redirects		`libcurl`
T.4	Connect in Browser (Puppeteer)	2024-04-12	Puppeteer
M.1	Query DNS Records	2024-04-12	JavaScript
M.2	Collect X.509 chains		
M.3	Fetch certificate from CT logs	2024-04-24	Go

TABLE C.2: List of DNS resource records that we collect for a given domain name. Subdomain column denotes the prefix added to the domain name for the query; @ denotes empty label.

#	Subdomain	Type	Description
1	@	SOA	Zone apex
2	@	A	IP address(es)
3	@	CAA	CAA RRs
4	_443._tcp	TLSA	DANE
5	_validation-contactemail	TXT	DV E-Mail contact
6	_validation-contactphone	TXT	DV Phone contact

Mismatch (CAA RRs does not match the cert issuer), and Issuer Match. Mismatching cases are further divided to denote if mismatch was caused by empty (`";"`) or malformed CAA `issue`/`issuewild` values. Implicit Match is semantically equivalent with Issuer Match but specifies the case where relevant CAA RR set is not empty yet has no `issue` or `issuewild` records.

To match if a certificate matches an issuer denoted by a CAA `issue`/`issuewild` record, we have composed our own approach that maps a CA string identifier to a CA based
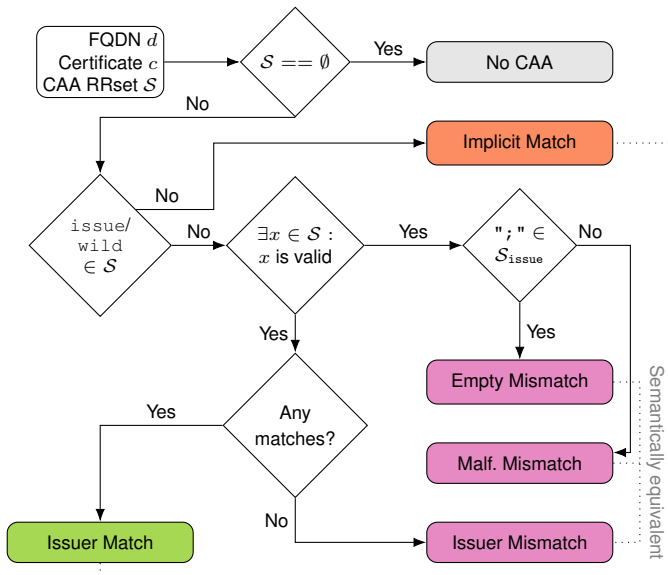
Fig. D.1: The algorithm we use to check the consistency between a certificate c and relevant CAA resource records set S for a given domain d.

on its subject name, organization name, or other features. For example, for the following CAA RR:

```
example.com          CAA 0 issue "web.com"
```

we assert that the CA certificate carries 'Network Solutions L.L.C.' as organization name in its subject field.