A Call to Reconsider Certification Authority Authorization (CAA)

Pouyan Fotouhi Tehrani, *TU Dresden, Germany*

Raphael Hiesgen, *HAW Hamburg, Germany*

Thomas C. Schmidt, *HAW Hamburg, Germany*

Matthias Wählisch, *TU Dresden, Germany*

Abstract—Certification Authority Authentication (CAA) is a safeguard against illegitimate certificate issuance. We show how shortcomings in CAA concepts and operational aspects undermine its effectiveness in preventing certificate misissuance. Our discussion reveals pitfalls and highlights best practices when designing security protocols based on DNS.

Keywords: Web PKI, DNS, CAA, Protocol Security

1. Introduction

Web security relies on X.509 certificates. Fundamental to the underlying security model is the correct issuance of certificates. This trust is challenged by unrestricted certification[1]. Certification Authorities (CA) are generally allowed to certify any arbitrary resource, *i.e.,* to bind a public key to a domain name by creating a certificate. If certification occurs without the consent of a name owner, the misissued certificate can be used to impersonate resources of that name owner. Causes for misissuance are manyfold. Rogue or compromised CAs[2], spoofed DNS records or compromised name servers[3], and malicious traffic rerouting[4,5] have been part of attack vectors in the past.

There are two principal directions to harden the security of the certificate issuance process. (*i*) *Prevention* by enabling a CA to verify whether the issuing request is valid. (*ii*) *Mitigation* by enabling the name owner or any third party on behalf to identify and revoke an incorrectly issued certificate. Both directions have been covered in standardization and deployment. Certificate transparency (CT)[6] makes certificates public and DNS-Based Authentication of Named Entities for TLS (DANE TLSA)[7] binds certificates to services available below a specific domain name. Prevention is specified in CA authorization (CAA)[8], which allows a name owner to denote in the DNS which CA is allowed to issue a certificate.

In this paper, we argue that both prevention and mitigation are important, but that prevention should be the first-class citizen because it addresses a root cause (§ 2). Unfortunately, CAA, which is not only an IETF standard but also the solution the CA/Browser Forum agreed on to be mandatory, is flawed. We revisit design decisions (§ 3) and deployment behaviors (§ 5) to refine on-going standards and inform our community to guide fundamentally different approaches (§ 6), hoping that these insights prevent common pitfalls in the future. Our data-driven approach (§ 4) is based on more than 4.6*M* unique certificates from CT logs, which we test whether they conform to CAA policies.

We show that there are serious concerns because CAA specification[8] and deployment agreements[9] suffer from four blind spots:

1) **Implicit semantics** mask misconfigurations and reduces expressiveness of CAA policies.
2) **Ambiguous CA identifiers** can (unintentionally) lead to overly permissive policies.
3) **Boundless policy scoping** allows parent zones or canonical name owners to impose policies on (unaware) child zones that lack explicit policies.
4) **Non-verifiable and temporally unrestricted policies** make CAA-based CA auditing unreliable if not impossible.

These four properties effectively defeat the primary purpose of CAA, which is to restrict issuance, as well as its secondary purpose, which is to enable *Evaluators* to audit CA issuance behavior, and increase security risks such as privilege escalation, name owner spoofing, and repudiation.

2. Certificate Authority Authorization

CAA[8] allows a domain name owner to specify which CA may issue certificates for the namespace under her control using dedicated `CAA` DNS resource

XXXX-XXX © IEEE
Digital Object Identifier 10.1109/XXX.0000.0000000

records (RR). Validating CAA was made mandatory for CAs by the CA/Browser Forum in 2017.

When a Certificate Signing Request (CSR) is presented to a CA, the CA must perform two steps. First, it must look up the appropriate `CAA` RRs for all domain names included. Second, the CA checks whether it is authorized to issue a certificate for the domain name based on the CAA policies it has retrieved.

The CAA lookup algorithm is iterative. If no `CAA` records are found for a given domain name (*e.g.,* www.example.com), the CA iteratively traverses the parent hierarchy (*e.g.,* example.com, then com) until `CAA` records are found or the root zone is reached. For aliases, the canonical name (denoted by a `CNAME` record) is first checked and if no `CAA` records are found, the parents of the alias are visited.

A CA can verify whether it is authorized to issue the requested certificate based on the CAA model, which provides properties (also called *tag*) and values as part of a CAA record. The IETF standardized `issue` and `issuewild` to constrain issuance and `iodef` for contacting the name owner in case of policy violations. The value of `issue` and `issuewild` contains an 'issuer-domain-name' to identify a CA and a list of optional parameters. CAs choose their own identifiers and list them in their Certification Practice Statement (CPS). It is possible to constrain issuance to multiple CAs or completely forbid issuance by having an `issue` or `issuewild` without any valid CA identifier. The `iodef` value must be a URL using `mailto`, `http`, or `https` as scheme.

The CAA specification allows non-standard tags to introduce new semantics and functionalities. Two additional tags are defined by the CA/Browser Forum to verify ownership of a name via phone (`contactphone`) or e-mail (`contactemail`). If a CA encounters an unsupported tag, it either rejects issuance if its critical flag is set, otherwise ignores the record. If only non-critical unknown tags or only tags other than `issue` or `issuewild` are found, it is interpreted as no constraint for issuance.

3. Threat Model

The CAA objective is to protect name owners from certificate misissuance. In the following we describe threats that undermine CAA effectiveness in securing certificate issuance. The focus of our threat model is on CAA design and deployment, such as authoritative DNS CAA records, or the certificate issuance process. Broader vulnerabilities of the DNS ecosystem, such as susceptibility to cache poisoning and record spoofing, or malicious or misbehaving CAs are out of scope.

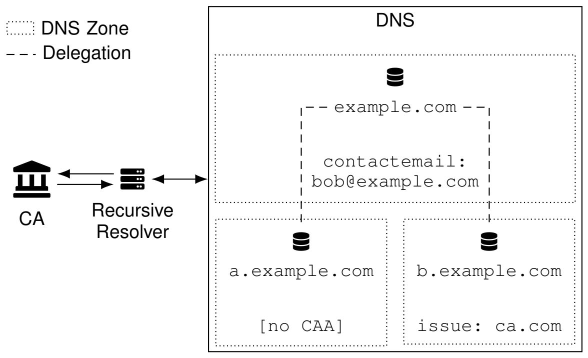

FIGURE 1. Overview of entities involved in CAA validation in a namespace with two delegations (`{a,b}.example.com`).

We assume that (*i*) target name servers and acquired DNS records are intact and authoritative, and (*ii*) CAs correctly implement and respect the CAA.

Elevation of privilege. Two issues with the concept and use of issuer-domain-name can lead to privilege escalation that authorizes an unintended CA. First, the CAA syntax is flexible enough for an actual misconfiguration to be interpreted as a valid policy, or different configurations can be used to express the same policy. For example, a typo in the CAA tag, *e.g.,* `isssue` instead of `issue`, can turn an intended CA authorization into a lifting of all restrictions. The other problem derives from an uncoordinated adoption of CA identifiers while the CAA specification does not require uniqueness or exclusive use of identifiers. Thus, an attacker may trick the name owner who is unaware of the scope of a policy into authorizing other CAs than intended (see Table 1). We discuss each of these issues in detail in § 5.

Spoofing name owner. The CAA allows policies to be defined across zone boundaries. This allows another entity, such as the owner of a canonical name or parent zone, to act on behalf of the actual name owner. In addition to defining issuance policies, if the other entity defines contact policies (*i.e.,* `contact[email|phone]`), it can use its own contact information for domain validation and acquire certificates for domain names that are effectively out of his control. For example, in Figure 1, the owner of `example.com` can apply for certificates for names within the `a.example.com` namespace and even pass domain validation even though that zone is delegated. Note that this threat is only viable if the domain in question does not define its own policies, which is enforced for aliases because DNS prohibits any record different from DNSSEC records next to a CNAME. This includes CAA records.

Name owner spoofing in this sense is based on the design decision to allow CAA policies to transcend DNS zone boundaries, and is fundamentally different from DNS name spoofing based on cache poisoning or record spoofing by abusing the lack of DNSSEC. We discuss these approaches in detail in § 5.3.

Repudiation. CAs document the issuance procedure together with relevant CAA records and are even mandated to do so in cases where issuance was prevented by a CAA record. However, in case of a misissuance, these records cannot be used to prove or refute wrongdoing by the CA as DNS records are neither signed nor carry a validity period by default. It is impossible to assert their authenticity or validity. In § 5.4 we discuss the need of verifiable and temporally bounded CAA policies in detail.

4. CAA Deployment in the Wild

We want to assess how the conceptional weaknesses of CAA reflect in operational practice. To measure CAA, we introduce a novel approach based on CT logs. Our measurements mimic what we expect from a CAA Evaluator[8]. Our method aims at (*i*) analyzing as many certificates as possible for each trusted CA, and (*ii*) minimizing the time discrepancy between certificate issuance and the collection of corresponding CAA records.

Certificate Collection. All major CAs submit their certificates to at least one CT log[10]. Compared to active TLS scans, scanning CT logs provide a more comprehensive picture of issued certificates by covering certificates which are not actively deployed on a server, *e.g.,* backup certificates. We scan all active CT logs compliant with the Chrome CT policy. In June 2024, 10 logs operated by 5 different companies were active. For each log, we regularly check for new precertificates and fetch these. We collect precertificates instead of final certificates since storing precertificates is mandatory whereas final certificates are optional.

Timely DNS queries. Maintaining the same set of CAA records as observed by CAs at the time of issuance is challenging because certificates are not instantly available in CT logs after submission. When a certificate is merged and publicly available depends on the maximum merge delay (MMD). Without access to historic DNS records, we argue that this remains the best effort in reconstructing those CAA records that the issuing CA observed.

We extract all DNS-type Subject Alternative Names (SANs) from merged certificates and expand each domain name to all of its parents up to TLDs. For example, `www.example.co.uk` is expanded to `{www.example.co.uk, example.co.uk, co.uk, uk}`. We then query CAA records for all names and store only certificates with at least one SAN that has a corresponding CAA record. The result is a mapping of domain names to a set of CAA records as well as several other attributes such as the source of record (*e.g.,* parent domain), provided the source is a canonical name.

To avoid stale data from third-party caches, we retrieve data directly from authoritative name servers instead of recursive resolvers.

Mapping between CA identifiers and CA names. To verify whether a certificate was issued by a legitimate CA, we need a mapping of CA identifiers (as denoted in CAA records) to CA names (as denoted in X.509 issuer field). Prior work[10,11] and the Mozilla Common CA Database provide such mappings. This data, however, is neither up-to-date nor complete. We extend the Mozilla list by inspecting CPS documents from various CAs.

Next, for each certificate, we consider all domain names included in the SAN and check whether CAA records comply with the CA that issued the certificate. Note that in contrast to prior work, our point of departure is certificates with *all* its included domain names, instead of a single domain name with its corresponding certificate deployed at a web server.

Measurement Setup and Data Corpus. We queried CT logs over a period of two week (2024-06-13 to 2024-06-26) and collected more than 5.4*M* certificates. About 85% (total of 4.6*M*) are unique certificates, because a single cert is logged in multiple logs as required by browsers. Among multiple instances of the same certificate, we choose the one that was first seen in any of the logs since it represents freshness. From these certificates, we extract 5.8*M* unique domains, of which 34% are wildcards.

Limitation. There is a temporal gap between when the CA queries DNS records and when we do. Changes to CA records during this window are not captured by our method. As shown later in § 5, even periods as short as 30 minutes are sufficient to skew the data. Regular monitoring, for example in a longitudinal study, is a viable extension of our method that can address this shortcoming.

Another challenge is correlating DNS zone ownership. Name owner spoofing, as described in § 3, is only a threat if policies are defined by separate entities. One approach could compare the `RNAME` (responsible person) value from the `SOA` records we query. In practice, however, this value usually indicates the entity responsible for the infrastructure, *e.g.,* DNS name server operator, rather than the name owner. Alternatives, such as querying DNS WHOIS databases, are also unreliable, as these typically redact personal information due to privacy concerns. Finally, we could use CAA contact information (*i.e.,* `contact[email|phone]`) for this purpose, but this information is rarely used in practice.

5. Potential Design and Operational Pitfalls

Here we place our data from § 4 in the context of the design choices that give rise to the threats introduced in § 3. We briefly discuss each design choice, explain its advantages, identify the associated problem, and propose solutions.

5.1. Implicit Semantics

Description. The CAA specification allows expressing different semantics using the same syntax or implementing the same semantic using different syntaxes. For example, `issue` can be used to (*i*) constrain issuance if its value is a CA-recognized domain name, (*ii*) forbid issuance if its value is empty (*i.e.,* `";"`), malformed (*e.g.,* `"%%%%%"`), or is not associated with any CA, or (*iii*) constrain issuance for wildcard names if `issuewild` is missing. Another example is the case of unrestricted issuance by omitting CAA records altogether *if* none of the parents declare CAA records, or by only listing a tag other than `issue` or `issuewild` with an arbitrary value. It is also allowed to have contradicting CAA policies in a CAA set. For example, to forbid issuance using an empty `issue` tag alongside of another `issue` tag with a valid value.

Advantage. This approach adds flexibility to the protocol. For example, if a name owner has the same policy for both FQDN and wildcard names, she can simply define a `issue` policy and omit an explicit `issuewild` policy with the same value. Beyond convenience, this also allows for smaller DNS packets and faster transmission.

Problem. First, the implicit semantic hinders implementing specific cases, such as defining constraints only for FQDN names, while posing no restrictions on wildcards. Second, it prevents CAs from differentiating between intentional configurations and accidental misconfiguration (*e.g.,* by typos).

Deployment. We observe 811 cases that permit certificate issuance and at the same time contain an empty `issue` tag. In additional 18 cases, malformed values coexist with valid identifiers. In other cases, we find the same erroneous, yet syntactically correct, values used at different domain names, indicating that the name owners have copied the values without realizing its semantics. For instance, we observe 1,319 unique domain names using `0issueletsencrypt.org` (incorrectly includes flag and tag within the value) with 829 using the following exact set to issue constraints: `{0issueletsencrypt.org, comodoca.com, digicert.com, letsencrypt.org, pki.goog}` (note the correct value for Let's Encrypt). Although the CAA specification explicitly ignores such cases[8], specifically for values that hint at a typo rather than an intentional malformed value, we diagnose that the semantic is not fully understood by name owners and can lead to unwanted results.

In our dataset, there are also cases of unintentional lifting of constraints due to unknown tags. Two domains use `isssue` tag (note extra "s") with a value of `letsencrypt.org`, which evidently was meant to constrain issuance, but in practice lifts all restrictions. Another domain uses the aforementioned erroneous tag alongside the correct `issue` tag. Similarly, there are cases where `issue` was misspelled (*e.g.,* `issed`) next to a valid `issuewild` records, thus only limiting issuance for wildcard domains. We also observe confusion by an issuer parsing the CA identifier only in `issuewild` but not the `issue` tag, leading the CA to consider itself to be authorized to certify FQDNs despite CAA constraints. We reported this incident to the CA (GoDaddy), which confirmed the failure (see Bugzilla #1904748).

Solution. First, the semantics of each tag should be uniquely confined, *i.e.,* `issue` is limited to FQDN domains, `issuewild` only applies to wildcard domains, and existence of (or lack thereof) one does not impact the other. Second, a constant value should be standardized to denote a no-issue policy instead of empty or malformed values. This way typos (as discussed below) or other unintentional errors are explicitly detectable and not open to misinterpretation. Finally, contradictory policies should be considered as errors.

5.2. Ambiguous Identifiers

Description. The CAA specification requires the 'issuer-domain-name' to be a valid domain name, but does not impose constraints on its ownership nor delegation status (*i.e.,* it can be a non-existent domain name). Domain names are used as identifiers to uniquely identify a CA in CAA policies.

Advantage. Domains are viable identifiers because uniqueness and potential ownership conflicts are resolved within the DNS ecosystem.

Problem. The way CA identifiers are used in practice does not guarantee uniqueness, since the same domain name will be accepted by any 'party acting under the explicit authority of the holder of the issuer domain name'[8]. This can be a subsidiary, a reseller, or even another CA organization. Ambiguous identifiers prevent name owners to authorize a specific CA. Two cases are plausible. First, a name owner unknowingly authorizes a third-party CA. This happens when independent CAs accept the same identifiers, *e.g.,* DigiCert also accepts Amazon identifiers as it vicariously runs some intermediate CAs for Amazon. Second, a name owner unknowingly authorizes all CAs running on the same infrastructure. This is the case for resellers, managed PKIs and alike. For example, DKB, a German bank, has its own dedicated CA managed by DigiCert, but lacks a dedicated identifier. Hence it uses `digicert.com`, which authorizes all DigiCert subsidiaries, such as Thawte or QuoVadis.

Deployment. Without additional information, it is not possible to detect whether name owners are aware of identifier ambiguities. In our dataset, nearly every fifth CAA source redundantly lists `comodoca.com` and `sectigo.com` for `issue` policy, whereas Sectigo CPS explicitly states that the former identifier is deprecated. We also observe up to eight different CA organizations that share the same identifier (see Table 1). Note that the number of CAs sharing identifiers is higher than what we observed in our measurements. Alone `sectigo.com` is accepted by Gandi (FR), ZeroSSL (CH), GEANT Vereniging (NL), eMudhra Technologies Limited (IN), and at least 17 other CAs from around the world that use Sectigo infrastructure.

Our data also revealed cases of misissuances (confirmed by the CA GoDaddy), in which all strings matching a pattern were accepted instead of a fixed domain name (see Bugzilla #1904749).

Solution. The identifiers should reference the CA with whom subscribers directly enter a business relationship instead of the infrastructure operator. Furthermore, an authoritative and publicly available mapping of CAA identifiers to CAs must be provided so that name owners can verify the scope of each identifier, and consider off-list values as invalid. Although Mozilla maintains such a mapping in its 'Common CA Database', it is neither authoritative nor complete[11]. The protocol registry of Internet Assigned Numbers Authority (IANA) is a potential alternative to maintain such a mapping, as was proposed for CAA tags[8]. Such mapping can be integrated into tooling to facilitate automation and feedback provision.

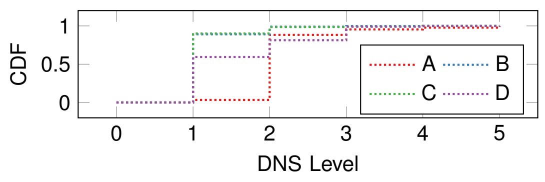

FIGURE 2. Distribution of domain names by DNS level at which relevant CAA records were found. Level 0 denotes that CAA records were found at the target domain name, level 1 at its direct parent, and so on. Group A are delegated domains w/o CAA, B are alias domains w/o CAA, C are alias parents w/ CAA, and D are alias domains w/o CAA but with an alias parent with CAA.

5.3. Boundless Policy Scoping

Description. The DNS namespace is partitioned in zones. Each zone comprises a set of authoritative data for all the names in its namespaces. A zone owner can further delegate parts of his namespace to others and create new zones. Delegated zones are independent of their parents so that the parent zone cannot define records for names in a delegated namespace without retracting the delegation. The CAA lookup algorithm breaks this authoritative barrier by allowing CAA records to be defined in a different authority, *i.e.,* be inherited from a parent zone or be defined by the canonical name owner in case of aliases.

Advantage. Inheritance allows policies to be defined once and applied to the entire (sub)namespace, simplifying administration. Furthermore, allowing the canonical name owner to define policies for an alias simplifies certificate application: since current practice allows the canonical name owner to obtain a certificate for the corresponding alias name (*e.g.,* through domain validation using the HTTP challenge), they might as well have control over CAA policies.

Problem. This expands the attack surface beyond the own zone, affecting not only issuance policies, but also contact information used for domain validation[9], *e.g.,* `contactemail`, which allows an entity to obtain a certificate for a domain name it does not control.

Deployment. More than 777*k* domain names in our dataset are subject to CAA policies defined by a separate zone. We classify these into four categories (ordered by frequency of occurrence):

A *Delegated domain w/o CAA.* In more than half

TABLE 1. The CA strings that appear in more than 10% of the CAA records in our dataset as well as their occurrence as the *only* CA string.

CAA String	`issue` Overall	`issue` Single	`issuewild` Overall	`issuewild` Single	CA Count[†]
`letsencrypt.org`	89.96%	37.62%	87.26%	5.68%	1
`digicert.com`	38.85%	0.40%	87.26%	0.93%	3
`pki.goog`	36.21%	5.90%	72.22%	8.46%	1
`comodoca.com`	27.84%	0.11%	67.16%	0.03%	6
`globalsign.com`	26.47%	0.02%	44.66%	0.12%	2
`sectigo.com`	25.00%	0.05%	33.48%	0.21%	8
`amazon.com`	11.24%	<0.01%	1.92%	0.04%	1
`godaddy.com`	8.25%	<0.01%	<0.01%	<0.01%	2

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

of all cases, the CAA is defined by a parenting zone for a subdomain that is delegated.

B *Alias domain w/o CAA.* About 45% of the cases show alias domains with CAA records sourced from a parent, *i.e.,* the authority of the domain name and CAA source differ.

C *Alias parent w/ CAA.* 1,672 instances inherit CAA and the relevant CAA records originate from a parenting domain name which is an alias for a domain name in another zone.

D *Alias domain w/o CAA and alias parent w/ CAA.* As a special instance of the former case, the domain name is itself an alias without CAA records, but we find a parent that is also an alias and its canonical name (from another zone) provides effective CAA records (1,080 instances).

In total, there were 108 domains for which a parent in another zone lists email addresses, so if the CA supports `contactemail`, the parent could apply and successfully retrieve a certificate for such a subdomain in another zone. In all instances it is another authority that defines CAA policies for a namespace. Figure 2 depicts the distribution of DNS hierarchy levels at which CAA records were found.

Solution. The effective scope of CAA records must be confined within a DNS zone. A naïve approach would adapt the lookup algorithm to stop at the zone apex[8]. This, however, does not solve the problem of aliases. The conceptually clean solution would be to use dedicated subdomains, similar to the DANE[7] approach for `TLSA` records, so that in the case of canonical names, the referencing authority has no control over the policy. For example, by using a designated `_caa` subdomain, *e.g.,* `_caa.example.com`.

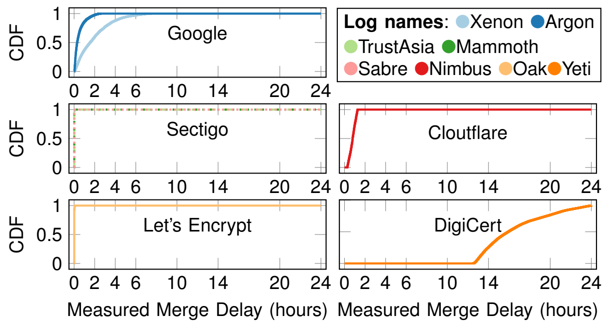

FIGURE 3. Distribution of measured CT merge delays.

5.4. Non-verifiable and Temporally Unbound Policies

Description. The CAA specification describes Evaluators as third-party auditors that may use CAA records to detect policy violations[8]. At the same time, it warns that DNS records may have changed at the time an Evaluator retrieves them and are effectively unreliable in discovering misissuances.

Advantage. CAA explicitly aims to be cost-effective through low barrier deployment requirements and simple validation processes[8]. Implementing non-repudiation or adding temporal validity constraints introduces additional overhead, both in terms of data and processing.

Problem. CAA can only be verified by the name owner and the CA at the time of certification. Even so, there is no definitive way for the CA to ensure that CAA records have not been tampered with or turned stale. Therefore, it is impossible to audit CAA and use DNS records to determine the root cause of certificate misissuance, whether it is CA misbehavior or name owner misconfiguration.

Deployment. To the best our knowledge, there are

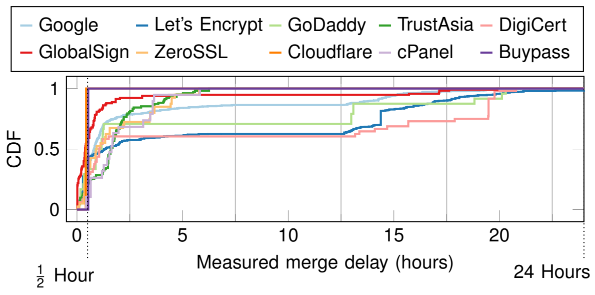

FIGURE 4. Distribution of measured merge delay for certificates that were incorrectly issued according to CAA records.

no publicly active CAA Evaluators. Using the measurements in this paper, we take the role of an Evaluator. Our proposed method in § 4 reduces the temporal gap between CA query of CAA records and our own queries. Nonetheless, a time discrepancy as large as the CT logs MMD (see § 4) remains. Figure 3 depicts this for various CT logs as measured by our scanners.

To understand the impact of time discrepancy we investigate cases of mismatching CAA policies in our dataset. Given that (*i*) we observe only a small fraction of certificates that mismatch their corresponding CAA records, and (*ii*) we are unable to reproduce such cases independently, we conjecture that these are false negatives due to modified DNS records. Figure 4 shows the distributions of these cases with respect to the measured merge delay. We can conclude that any time discrepancy larger than 30 minutes significantly increases the chance of incorrect CAA validation. Regardless of merge delays, CAs can also use CAA records as stale as 8 hours[9], so that in the worst case the temporal discrepancy amounts to $8h + MMD$. Even Sunlight logs, a new generation of CT logs that remove merge delays, cannot fill this gap. Without fine-granular historical DNS data or designated validity timestamps for CAA records, auditing of CAA remains unreliable.

Solution. This problem can be addressed indirectly by enabling DNSSEC, or directly by adding validity timestamps to CAA records and signing them. The former approach infers validity of a record from validity period of covering DNSSEC signatures, while data-origins can be validated using DNSSEC keys for that zone. In the latter approach CAA records are extended to (*i*) carry a validity timestamp, and (*ii*) be cryptographically secured, *e.g.,* through a digital signature. This also allows to determine which entity defined a set of CAA policies (data-origin authentication). A signed and time-constrained CAA record can be verified retrospectively and securely, *e.g.,* by an Evaluator, but requires mechanisms to distribute keys.

6. Lessons learned

CA Authorization was designed as a preventive measure against certificate misissuance but falls short due to design decisions, which hinder proper deployment. We highlighted pitfalls and proposed solutions. In the following, we report about lessons learned from a decade of CAA usage and discuss aspects that should guide the design of related protocols in the future.

DNS is a fragile basis for security protocols. Any service that relies on the unprotected DNS inherits its vulnerabilities. For example, it has been shown how plain DNS can impact domain validation and consequently lead to certificate misissuance[3]. Although aware of this issue, CAA does not provide an alternative and only recommends using DNSSEC —without making it mandatory—to mitigate threats targeting DNS[8]. DNSSEC, a set of security extensions, brings integrity and origin authentication alongside authentication of name nonexistence to DNS. In the context of the Web, however, DNSSEC is not widely used by name servers and resolvers[12–14].Nevertheless, its viability and success can be observed within the email ecosystem.

Solutions orthogonal to DNSSEC have been developed to mitigate DNS security weaknesses, among them Multi-Perspective Issuance Corroboration (MPIC), and trust on first use (TOFU). MPIC verifies domain control from multiple vantage points across the Internet to reduce the risk of retrieving manipulated data. It has been integrated into CA/Browser Forum Baseline Requirements as part of domain validation procedures. TOFU-based approaches rely on a first intact and uncompromised data exchange to establish security parameters for upcomming interactions. An example is MTA-STS, which is used against downgrade attacks in STARTTLS protocols (*e.g.,* deployed by mail servers).

Trust must be derived from frequent public audits. To measure the effectiveness of a security protocol, it must be auditable. Experiences from the Web PKI show that formal and private audits (*e.g.,* WebTrust or ETSI) are rather blunt tools for preventing or detecting certificate missiuances[2]. In contrast, providing the public with adequate means to perform audits is effective as the success story of CT logs manifests.

CAA validation by CAs cannot be audited. CAs do not publicly log CAA record consumption nor even provide them upon request[11], and even if they did, they could not rule out manipulation or spoofing. For DNS-based protocols to be auditable, some form of content object security is required for the underlying resource records to be temporally and spatially decoupled from the name owner—as provided by DNSSEC. Once object security is reached for DNS resource records, decoupled data can be distributed over established channels, for example as extensions to the Online Certificate Status Protocol (OCSP) or TLS (*cf.,* TLS DNSSEC Chain Extension).

Potential victims must be able to verify measures. CAA has been designed as a light-weight approach to protect against certificate misissuance. It enables the CAs—and only the CAs—to verify their procedures at the time of certificate issuance.

Potential victims of illegitimate certificates, however, are resource owners and service users. An online financial institution, for example, looses reputation if its customers get fooled by an illegitimately certified fraud service. Neither the legitimate service provider nor its users can reliably detect cases of misissuance, which leaves them blind with respect to the efficacy of the CAA protective measures. This blindness of potential victims turns CAA into a toothless security scheme. Instead, CAA merely serves as a detection aid for configuration errors in toolchains of well-behaving CAs.

7. Conclusions and Outlook

A reliable and trustworthy certificate issuing process is utmost important to enable secure Internet services. In this paper, we critically revisited CAA, a currently popular DNS-based Internet standard for preventing illegitimate issuance of certificates. The CA/Browser Forum mandates the deployment of CAA. We found the design of CAA flawed to an extent that prevents proper deployment and proper auditing. Given the importance of the security objectives and that some problems have been found in the wild already in 2018, the lack of progress leaves us puzzled.

We identified several options that require only minor changes to improve CAA in future development, and highlighted principle design choices that should guide the design of improvements. With these insights we hope to have shed light on how to build a more trustworthy certificate ecosystem in the near future.

Ethical Concerns. This position paper draws attention to a security approach that should be improved with respect to design and deployment. When we found incidents in real deployments, we implemented a responsible disclosure policy to solve the problem with the concerned CA before making the incidents public.

8. Acknowledgments

This work was supported in parts by the German Federal Ministry of Education and Research (BMBF) within the project PRIMEnet.

REFERENCES

1. P. F. Tehrani, E. Osterweil, T. C. Schmidt, and M. Wählisch, “A Security Model for Web-Based Communication,” *Communication of the ACM*, 2024. [Online]. Available: https://dl.acm.org/doi/10.0.4.121/3623292
2. 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, pp. 325–340. [Online]. Available: https://doi.org/10.1145/3131365.3131401
3. 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*. Piscataway, NJ, USA: IEEE, 2019, pp. 544–559. [Online]. Available: https://doi.org/10.1109/EuroSP.2019.00046
4. A. Gavrichenkov, “Breaking HTTPS with BGP Hijacking,” website, 2015, black Hat. Briefings. [Online]. Available: https://www.blackhat.com/us-15/briefings.html
5. H. Birge-Lee, Y. Sun, A. Edmundson, J. Rexford, and P. Mittal, “Bamboozling certificate authorities with BGP,” in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 833–849. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/birge-lee
6. 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
7. 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
8. 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
9. 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
10. P. F. Tehrani, R. Hiesgen, T. Lübeck, T. C.

Schmidt, and M. Wählisch, "Do CAA, CT, and DANE Interlink in Certificate Deployments? A Web PKI Measurement Study," in *Proc. of Network Traffic Measurement and Analysis Conference (TMA)*. Piscataway, NJ, USA: IEEE Press, May 2024, pp. 1–11. [Online]. Available: https://doi.org/10.23919/TMA62044.2024.10559089

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. E. Osterweil, M. Ryan, D. Massey, and L. Zhang, "Quantifying the operational status of the dnssec deployment," in *Proc. of the 8th ACM SIGCOMM Conference on Internet Measurement*. New York, NY, USA: ACM, 2008, pp. 231–242.
13. T. Chung, R. Van Rijswijk-Deij, B. Chandrasekaran, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "A longitudinal, end-to-end view of the DNSSEC ecosystem," in *Proc. of the 26th USENIX Security Symposium*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 1307–1322.
14. E. Osterweil, P. F. Tehrani, T. C. Schmidt, and M. Wählisch, "From the Beginning: Key Transitions in the First 15 Years of DNSSEC," *Transactions on Network and Service Management (TNSM)*, vol. 19, no. 4, pp. 5265–5283, December 2022. [Online]. Available: https://doi.org/10.1109/TNSM.2022.3195406

Pouyan Fotouhi Tehrani is a final-year PhD student and a research associate in the Chair of Distributed and Networked Systems at TU Dresden. His area of interest is secure and trustworthy Internet communication, covering both technical and non-technical aspects. This includes research on public key infrastructures (Web PKI and DNNSEC) and related Web protocols, the DNS ecosystem, large-scale Internet measurements, and network dependency analysis.

Raphael Hiesgen is currently pursuing a Ph.D. at HAW Hamburg under the guidance of Prof. T. C. Schmidt and Prof. Wählisch. His research centers on Internet measurements, specifically in the context of security applications. Proficient in C++ and adept at crafting scalable systems, he developed Spoki, a reactive network telescope tailored for long-term measurements. His skill set spans from designing and implementing measurement systems to delving into the analysis of collected data. Spoki was initially presented at USENIX Security in 2022 and is undergoing continuous improvements to broaden its visibility into the scanning ecosystem.

Thomas C. Schmidt Thomas C. Schmidt is a Professor of computer networks and Internet technologies with the Hamburg University of Applied Sciences, Hamburg, Germany, where he heads the Internet Technologies Research Group. He was the Principal Investigator in a number of EU, nationally funded, and industrial projects, as well as a Visiting Professor with the University of Reading, Reading, U.K. He is a co-founder and a coordinator of several successful open source communities. His current research interests include development, measurement, and analysis of large-scale distributed systems, such as the Internet or its offsprings.

Matthias Wählisch Matthias Wählisch holds the Chair of Distributed and Networked Systems at TU Dresden, and is a Research Fellow of the Barkhausen Institut. His research and teaching focus on scalable, reliable, and secure Internet communication. This includes the design and evaluation of networking protocols and architectures, as well as Internet measurements and analysis. Matthias is involved in the IETF since 2005 and co-founded multiple successful open source projects.