Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: 3 September 2026

S. Sahib
Brave Software
S. Huque
Salesforce
P. Wouters
Aiven
E. Nygren
Akamai Technologies
T. Wicinski
Cox Communications

2 March 2026

# Domain Control Validation using DNS
## draft-ietf-dnsop-domain-verification-techniques-12

## Abstract

Many application services on the Internet need to verify ownership or control of a domain in the Domain Name System (DNS). The general term for this process is "Domain Control Validation", and can be done using a variety of methods such as email, HTTP/HTTPS, or the DNS itself. This document focuses only on DNS-based methods, which typically involve the Application Service Provider requesting a DNS record with a specific format and content to be visible in the domain to be verified. There is wide variation in the details of these methods today. This document provides some best practices to avoid known problems.

## Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/ietf-wg-dnsop/draft-ietf-dnsop-domain-verification-techniques/.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

## Copyright Notice



## Table of Contents

## 1. Introduction

Many Application Service Providers of internet services need domain owners to prove that they control a particular DNS domain before the Application Service Provider can operate services for or grant some privilege to that domain. For instance, Certification Authorities (CAs) ask requesters of TLS certificates to prove that they operate the domain they are requesting the certificate for. Application Service Providers generally allow for several different ways of proving control of a domain. In practice, DNS-based methods take the form of the Application Service Provider generating a Unique Token and asking the requester to create a DNS record containing this Unique Token and placing it at a location within the domain that the Application Service Provider can query for.

This document recommends using a TXT based DNS Validation Record in a way that is targeted to the specific application service, and uses Unique Tokens to guarantee uniqueness.

## 2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

* Application Service Provider: an internet-based provider of a service, for e.g., a Certification Authority or a service that allows for user-controlled websites. These services often require

a User to verify that they control a domain. The Application Service Provider may be implementing a standard protocol for domain validation (such as [RFC8555]) or they may have their own specification.

* DNS Administrator: the owner or responsible party for the contents of a domain in the DNS.

* Intermediary: an internet-based service that leverages the services of other providers on behalf of a User. For example, an Intermediary might be a service that allows for User-controlled websites and in-turn needs to use a Certification Authority provider to get TLS certificates for the User on behalf of the website.

* User: the owner or operator of a domain in the DNS who needs to prove ownership of that domain to an Application Service Provider, often on behalf of an account at the Application Service Provider, working in coordination with their DNS Administrator.

* Unique Token: a value that uniquely identifies the DNS domain control validation challenge, defined in Section 6.1.1. Unique Tokens are constructed by the Application Service Provider in a way that guarantees uniqueness within the scope of the challenge, such as a random value.

* Validation Record: the DNS record that is used to prove ownership of a domain name ([RFC9499]). It typically contains an unguessable value generated by the Application Service Provider which serves as the DNS challenge. The Application Service Provider looks for the Validation Record in the zone of the domain name being verified and checks if it contains the unguessable value.

## 3. Purpose of Domain Control Validation

Domain Control Validation allows a User to demonstrate to an Application Service Provider that they have enough control over a domain to place a DNS challenge provided by Application Service Provider into the domain. Because this challenge becomes publically visible as soon as it is published into the DNS, the security properties rely on the causal relationship between the Application Service Provider generating a specific challenge and the challenge appearing in the DNS at a specified location. Domain Control Validation can be used either as a one-off or for a persistent validation depending on the application scenario:

* As a one-off validation, the Validation Record is time-bound, and it can be removed once its presence is confirmed by the Application Service Provider. These are appropriate when the validation is being performed as part of an action such as requesting certificate issuance.

* As a persistent validation, the introduction of the Validation Record into the domain demonstrates to the Application Service Provider that the User had control over the domain at that time, and its continued presence demonstrates only that either the DNS Administrator of the domain has left the Validation Record in-place (perhaps unintentionally) or that a new owner of the domain has re-introduced the Validation Record. The validation can be revoked by removing the Validation Record although this revocation will not be noticed until the Application Service Provider next checks for the presence of the record.

Persistent validation is only appropriate for applications where the validation is tightly coupled to the User at the Application Service Provider, as once a token is disclosed there is no guarantee that it hasn’t been copied by the new owner of a domain.

Delegated Domain Validation (Section 7) is a method typically used as a way to adapt between these modes, with a persistent validation to an Intermediary enabling the Intermediary to transitively perform recurring one-off validations.

## 4. Threat Model

As Domain Control Verification is a mechanism trying to provide security properties over sometimes-insecure underlying protocols, it is important to be clear about both its threat model.

While the specific primary Unacceptable Losses will depend on the nature of the Application Service Provider, they generalize to:

* UL1. Application Service Provider believes a User has privileges on a domain name without this being authorized by the DNS Administrator for the domain. The Threat Actor in this case is a malicious User leveraging these privileges in some way.

* UL2. Application Service Provider, Intermediary, or other party gains unintended control over resources within a domain or on a domain name. The Threat Actor in this case is the Application Service Provider, Intermediary, or other party leveraging this unintended control in some way.

### 4.1. Hazards leading to Unauthorized Priviledges (UL1)

For UL1, the Application-specific nature of these priviledges (such as being able to obtain a signed certificate covering the domain name, being able to use a social media handle under that domain, or being able to provision configurations associated with that domain int the Application Service Provider system) will determine the specifics of the underlying Unacceptable Loss.

Domain Control Validation attempts to address UL1 by having the User demonstrate relationship between the Application Service Provider issuing a Unique Token and that Unique Token appearing in domain. Classes of Hazards include:

* H1. Unique Token collision leading to an unassociated but matching Validation Record already being present in the domain, thus violating the causality property.

* H2. Cross-User vulnerabilities leading to a Unique Token issued to one User being leveraged by a different User, due to vulnerabilities in how an Application Service Provider or Intermediary implements Domain Control Validation.

* H3. Network and DNS based attacks leading to a Application Service Provider’s validation system being tricked into believing that a valid Validation Record containing the Unique Token is present. When DNS resolutions are not authenticated, this may be due to on-path network attackers, network attackers inserting themselves on-path (e.g., [RFC7132]), or other DNS protocol attacks (see [RFC3833].

* H4. DNS Administrator errors, including human factor issues, leading to a Validation Record being unintentionally added or unintentionally persisting.

* H5. Confusion over the scope of a Validation Record resulting in broader privileges being granted to the User than was intended by the DNS Administrator. This is discussed more below in Section 5.

### 4.2. Hazards leading to Unintended Access to Domain Resources (UL2)

For UL2, unintended control over a domain or domain name results as a side-effect of the Domain Control Validation process itself. Classes of Hazards include:

* H6. The owner name of the Validation Record is meaningful in other contexts, enabling cross-protocol, privilege escalation, and/or confused deputy attacks. For example, if a Validation

Record is a CNAME and has an owner name that is a valid hostname, the Application Service Provider could provide services on the Validation Record name within the domain.

## 5. Scope of Validation

For security reasons (see H5 in Section 4.1), it is crucial to understand the scope of the domain name being validated. Both Application Service Providers and the User need to clearly specify and understand whether the validation request is for a single hostname, a wildcard (all hostnames immediately under that domain), or for the entire domain and subdomains rooted at that name. This is particularly important in large multi-tenant enterprises, where an individual deployer of a service may not necessarily have operational authority of an entire domain.

In the case of X.509 certificate issuance, the certificate signing request and associated challenge are clear about whether they are for a single host or a wildcard domain. Unfortunately, the ACME protocol's DNS-01 challenge mechanism ([RFC8555], Section 8.4) does not differentiate these cases in the DNS Validation Record. In the absence of this distinction, the DNS administrator tasked with deploying the Validation Record may need to explicitly confirm the details of the certificate issuance request to make sure the certificate is not given broader authority than the User intended.

In the more general case of an Internet application service granting authority to a domain owner, again no existing DNS challenge scheme makes this distinction today. New applications should consider having different application names for different scopes, as described. Regardless, services should very clearly indicate the scope of the validation in their public documentation so that the domain administrator can use this information to assess whether the Validation Record is granting the appropriately scoped authority.

## 6. Recommendations

All Domain Control Validation mechanisms are implemented by a DNS resource record with at least the following information:

1. A record name related to the domain name being validated, usually constructed by prepending an application specific label.

2. One or more Unique Tokens.

### 6.1. TXT Record based Validation

The RECOMMENDED method of doing DNS-based domain control validation is to use DNS TXT records as the Validation Record. The QNAME is constructed as described in Section 6.2, and the RDATA MUST contain at least a Unique Token provided by the Application Service Provider (constructed according to the properties described in Section 6.1.1). If there are multiple character-strings within the RDATA, the Application Service Provider MUST treat them as a concatenated string. If metadata (see Section 6.1.2) is not used, then the Unique Token generated as-above can be placed as the only contents of the RDATA. For example:

```
_example_service-challenge.example.com.  IN   TXT  "3419...3d206c4"
```

This again allows the Application Service Provider to query only for application-specific records it needs, while giving flexibility to the User adding the DNS record (i.e., they can be given permission to only add records under a specific prefix by the DNS administrator).

Application Service Providers MUST validate that a Unique Token in the TXT record matches the one that they gave to the User for that specific domain name. Whether multiple Validation Records can exist for the same domain is up to the Application Service Provider's application specification. In case there are multiple TXT records for the specific domain name, the Application Service Provider MUST confirm at least one record match.

#### 6.1.1. Unique Token

A Unique Token is used in the challenge and is a value issued between parties (Application Service Provider to User, Application Service Provider to Intermediary, or Intermediary to User). The Unique Token MUST be constructed in a manner which has adequate uniqueness so as to guarantee a causal relationship between its issuance and its appearance in a DNS record. If multiple Application Service Providers are using the same Validation Record name then the Unique Token MUST be constructed in a way that prevents collisions.

Examples of Unique Token construction include:

* A random token, such as constructed according to Section 6.1.1.1
* A URI [RFC3986] namespaced to the Application Service Provider and uniquely identifying the challenge or User
* A keyed cryptographic hash of information known to the Application Service Provider which uniquely identifies the challenge or User

This Unique Token is placed in either the RDATA or an owner name, as described in the rest of this section. Some methods of validation may involve multiple independent Unique Tokens.

If sensitive information is used to derive a Unique Token, that information should be fed through a potentially keyed cryptographic hash as part of constructing the token.

Base32 encoding ([RFC4648], Section 6) or hexadecimal base16 encoding ([RFC4648], Section 8) are RECOMMENDED to be specified when the Unique Token would exist in a DNS label such as in a CNAME target. This is because base64 relies on mixed case (and DNS is case-insensitive as clarified in [RFC4343]) and because some base64 characters ("/", "+", and "=") may not be permitted by implementations that limit allowed characters to those allowed in hostnames. If base32 is used, it SHOULD be specified in way that safely omits the trailing padding ("="). Note that DNS labels are limited to 63 octets which limits how large such a token may be.

#### 6.1.1.1. Random Token Construction

One way of constructing Unique Tokens is to use random values which:

1. have adequate entropy to guarantee uniqueness and ensure that an attacker is unable to create a situation where a collision occurs (see H1 in Section 4.1).

2. are base64url ([RFC4648], Section 5) encoded, base32 encoded, or hexadecimal base16 encoded.

### 6.1.2. Token Metadata

It may be desirable to associate metadata with the Unique Token in a Validation Record. When specified, metadata SHOULD be encoded in the RDATA via space-separated ASCII key-value pairs, with the key "token" prefixing the Unique Token. For example:

```
_example_service-challenge.example.com.  IN   TXT  "token=3419...3d206c4"
```

If there are multiple tokens required, each one MUST be in a separate RR to allow them to match up with any additional attributes. For example:

```
_example_service-challenge.example.com.  IN   TXT  "token=3419...3d206c4 attr=bar"
                              IN   TXT  "token=5454...45dc45a attr=quux"
```

The token MUST be the first element in the key-value list. If the TXT record RDATA is not prefixed with token= then the entire RDATA should be assumed to be the token (as this might split the trailing "==" or "=" at the end of base64 encoding).

Keys are considered to be case-insensitive. Each Validation Record consists of RDATA for val-record with the following grammar (with an ABNF per [RFC5234]):

```
val-record     = keyvalue-list
keyvalue-list  = keyvalue-pair *( SP keyvalue-pair )
keyvalue-pair  = key [ "=" value ]

key            = 1*key-char
key-char       = ALPHA / DIGIT / "-" / "_"

value          = 1*value-char
value-char     = value-char = %x21-21 / %x23-5B / %x5D-7E
                 ; All printable ASCII except space (0x20),
                 ; quotation mark (0x22), and backslash (0x5C)
```

If an alternate syntax is used by the Application Service Provider for token metadata, they MUST specify a grammar for it.

## 6.2. Validation Record Owner Name

The RECOMMENDED format for a Validation Record’s owner name is application-specific underscore prefix labels. Domain Control Validation Records are constructed by the Application Service Provider by prepending the label "_<PROVIDER_RELEVANT_NAME>-challenge" to the domain name being validated (e.g. "_example_service-challenge.example.com"). The prefix "_" is used to avoid collisions with existing hostnames and to prevent the owner name from being a valid hostname (see H6 in Section 4.2).

If an Application Service Provider has an application-specific need to have multiple validations for the same label, multiple prefixes can be used, such as "_<FEATURE>._<PROVIDER_RELEVANT_NAME>-challenge".

Application owners SHOULD utilize the IANA "Underscored and Globally Scoped DNS Node Names" registry [UNDERSCORE-REGISTRY] and avoid using underscore labels that already exist in the registry.

As a simplification, some applications may decide to omit the "-challenge" suffix and use just "_<PROVIDER_RELEVANT_NAME>" as the label.

### 6.3. Time-bound checking and Expiration

For persistent validations, Application Service Providers MUST provide clear instructions for how to perform revocations through the removal of a Validation Record, including details on the frequency at which re-validation is performed. Application Service Providers MAY monitor for changes in domain ownership and request re-confirmation via a new token.

For one-off validations, after domain control validation is completed there is typically no need for the Validation Record to continue to exist after being confirmed by the Application Service Provider. It should be safe to remove the validation DNS record once the validation is complete.

Application Service Providers MUST provide clear instructions on how long the challenge token is valid for, and thus when a Validation Record can be removed. These instructions should preferably be encoded within the RDATA.

The instructions for validity duration MAY be encoded in the RDATA as token metadata (Section 6.1.2 using the key "expiry" to hold a time after which it is safe to remove the Validation Record. For example:

```
_example_service-challenge.example.com.  IN   TXT  "token=3419...3d206c4 expiry=2023-02-0
8"
```

When an expiry time is specified, the value of "expiry" SHALL be in ISO 8601 format as specified in [RFC3339], Section 5.6.

Alternatively, if the record should never expire (for instance, persistent validations that are checked periodically by the Application Service Provider) and should not be removed, the "expiry" key SHALL be set as "expiry=never".

The "expiry" key MAY be omitted in cases where the Application Service Provider has clarified the record expiry policy out-of-band. In this case, the RDATA is set to "token=3419...3d206c4". This is semantically identical to "3419...3d206c4".

The User SHOULD de-provision the resource record provisioned for DNS-based domain control validation once it is no longer required.

### 6.4. TTL Considerations

The TTL [RFC1034] for Validation Records SHOULD be short to allow recovering from potential misconfigurations. These records will not be polled frequently so expected caching or resolver load will be limited during normal operations.

The Application Service Provider looking up a Validation Record may have to wait for up to the SOA minimum TTL (negative caching TTL) of the enclosing zone for the record to become visible, if it has been previously queried. If the application User wants to make the Validation Record visible more quickly they may need to work with the DNS administrator to see if they are willing to lower the SOA minimum TTL (which has implications across the entire zone).

Application Service Providers' verifiers MAY wish to use dedicated DNS resolvers configured with a low maximum negative caching TTL, flush Validation Records from resolver caches prior to issuing queries or just directly query authoritative name servers to avoid caching.

## 7. Delegated Domain Control Validation

Delegated domain control validation lets a User delegate the domain control validation process for their domain to an Intermediary without granting the Intermediary the ability to make changes to their domain or zone configuration. It is a variation of TXT record validation (Section 6.1) that indirectly inserts a CNAME record prior to the TXT record.

The Intermediary gives the User a CNAME record to add for the domain and Application Service Provider being validated that points to the Intermediary's domain, where the actual validation TXT record is placed. The canonical name in the CNAME record is constructed as a base16-encoded (or base32-encoded) Intermediary Unique Token (generated as in Section 6.1.1) prefixed onto a domain operated by the Intermediary. For example:

```
_example_service-challenge.example.com. IN CNAME <intermediary-unique-token>.dcv.inte
rmediary.example.
```

The Intermediary then adds the actual Validation Record in a domain they control:

```
<intermediary-unique-token>.dcv.intermediary.example. IN TXT "<provider-unique-token>"
```

Such a setup is especially useful when the Application Service Provider wants to periodically re-issue the challenge with a new provider Unique Token. CNAMEs allow automating the renewal process by letting the Intermediary place the Unique Token in their DNS zone instead of needing continuous write access to the User's DNS.

Importantly, the CNAME record target also contains a Unique Token issued by the Intermediary to the User (preferably over a secure channel) which proves to the Intermediary that example.com is controlled by the User (see H2 in Section 4.1). The Intermediary

must keep an association of Users and domain names to the associated Intermediary-Unique-Tokens. Without a linkage validated by the Intermediary during provisioning and renewal there is the risk that an attacker could leverage a "dangling CNAME" to perform a "subdomain takeover" attack ([SUBDOMAIN-TAKEOVER]).

When a User stops using the Intermediary they should remove the domain control validation CNAME in addition to any other records they have associated with the Intermediary.

## 8. Supporting Multiple Accounts and Multiple Intermediaries

There are use-cases where a User may wish to simultaneously use multiple intermediaries or multiple independent accounts with an Application Service Provider. For example, a hostname may be using a "multi-CDN" where the hostname simultaneously uses multiple Content Delivery Network (CDN) providers.

To support this, Application Service Providers may support prefixing the challenge with a label containing an unique account identifier of the form _<identifier-unique-token>. The identifier-unique-token is a base16-encoded (or base32-encoded) Unique Token (generated as in Section 6.1.1. If the identifier is sensitive in nature, it should be run through a truncated hashing algorithm first. The identifier token should be stable over time and would be provided to the User by the Application Service Provider, or by an Intermediary in the case where domain validation is delegated (Section 7).

The resulting record could either directly contain a TXT record or a CNAME (as in Section 7). For example:

```
_<identifier-unique-token>._example_service-challenge.example.com.  IN   TXT  "3419...3d2
06c4"
```

or

```
_<identifier-unique-token>._example_service-challenge.example.com.  IN   CNAME  <intermed
iary-random-token>.dcv.intermediary.example.
```

When performing validation, the Application Service Provider would resolve the DNS name containing the appropriate identifier unique token.

The ACME protocol has incorporated this method to specify DNS account specific challenges in [ACME-DNS-ACCOUNT-LABEL].

Application Service Providers may wish to always prepend the _<identifier-token> to make it harder for third parties to scan, even absent supporting multiple intermediaries. The _<identifier-token> MUST start with an underscore so as to not be a valid hostname (see H6 in Section 4.2).

# 9. Security Considerations

## 9.1. Token Collisions

If token values aren’t long enough, lack adequate entropy, or are not unique there’s a risk that a malicious actor could obtain a token that collides with one already present in a domain through repeated attempts (H1 in Section 4.1).

Application Service Providers MUST evaluate the threat model for their particular application to determine a token construction mechanism that guarantees uniqueness and meets their security requirements (UL1 in Section 4).

When Random Tokens are used, they MUST be constructed in a way that provides sufficient unpredictability to avoid collisions and brute force attacks.

## 9.2. Token Confusion

If token values in challenge labels (Section 8) aren’t long enough or lack adequate entropy there’s a risk that a malicious actor could produce a token that could be confused with an application-specific underscore prefix label (H6 in Section 4.2).

## 9.3. Service Confusion

A malicious Application Service Provider that promises to deliver something after domain control validation could surreptitiously ask another Application Service Provider to start processing or sending mail for the target domain and then present the victim User with this DNS TXT record pretending to be for their service. Once the User has added the DNS TXT record, instead of getting their service, their domain is now certifying another service of which they are not aware they are now a consumer. If services use a clear description and name attribution in the required DNS TXT record, this can be avoided. For example, by requiring a DNS TXT record at _vendorname.example.com instead of at example.com, a malicious service could no longer forward a challenge from a different service without the User noticing. Both the Application Service Provider and the service being authenticated and authorized should be unambiguous from the Validation Record to prevent malicious services from misleading the

domain owner into certifying a different provider or service. (H2, H4, H5, and H6 in Section 4)

### 9.4. Service Collision

As a corollary to Section 9.3, if the Validation Record is not well-scoped and unambiguous with respect to the Application Service Provider, it could be used to authorize use of another Application Service Provider or service in addition to the original Application Service Provider or service. (H2, H4, H5, and H6 in Section 4)

### 9.5. Scope Confusion

Ambiguity of scope introduces risks, as described in Section 5. Distinguishing the scope in the application-specific label, along with good documentation, should help make it clear to DNS administrators whether the record applies to a single hostname, a wildcard, or an entire domain. Always using this indication rather than having a default scope reduces ambiguity, especially for protocols that may have used a shared application-specific label for different scopes in the past. While it would also have been possible to include the scope as an attribute in the TXT record, that has more potential for ambiguity and misleading an operator, such as if an implementation ignores an attribute it doesn’t recognize but an attacker includes the attribute to mislead the DNS administrator. (H5 in Section 4)

### 9.6. Authenticated Channels

Application Service Providers and intermediaries should use authenticated channels to convey instructions and Unique Tokens to Users. Otherwise, an attacker in the middle could alter the instructions, potentially allowing the attacker to provision the service instead of the User. (H3 in Section 4.1)

### 9.7. DNS Spoofing and DNSSEC Validation

A domain owner SHOULD sign their DNS zone using DNSSEC [RFC9364] to protect Validation Records against DNS spoofing attacks, including from on-path attackers.

Application Service Providers MUST use a trusted DNSSEC validating resolver to verify Validation Records they have requested to be deployed. When the AD bit ([RFC4035] Section 3.2.3) is not set in DNS responses for Validation Records, Application Service Providers SHOULD take additional steps to reduce an attacker’s ability to complete a challenge by spoofing DNS:

* Application Service Providers SHOULD attempt to query and confirm the Validation Record by matching responses from multiple DNS resolvers on unpredictable geographically diverse IP addresses

* Application Service Providers MAY perform multiple queries spread out over a longer time period to reduce the chance of receiving spoofed DNS answers.

DNS Spoofing attacks are easier in the case of persistent validation as the expected result is publicly known. For example, absent DNSSEC this could allow an on-path attacker to bypass a revocation by continuing to return a record that the DNS Operator had removed from the zone.

The above are needed to address H3 in Section 4.1.

### 9.8. Application Usage Enumeration

The presence of a Validation Record with a predictable domain name (either as a TXT record for the exact domain name where control is being validated or with a well-known label) can allow attackers to enumerate the utilized set of Application Service Providers. The use of Section 8 can make it harder to scan if the identifier-unique-token is long enough, but can also expose User account information depending on how the identifier-unique-token is encoded.

### 9.9. Public Suffixes

As discussed in Appendix A.2, there are risks in allowing control to be demonstrated over domains which are "public suffixes" (such as ".co.uk" or ".com"). The volunteer-managed Public Suffix List ([PSL]) is one mechanism that can be used. It includes two "divisions" ([PSL-DIVISIONS]) covering both registry-owned public suffixes (the "ICANN" division) and a "PRIVATE" division covering domains submitted by the domain owner.

Operators of domains which are in the "PRIVATE" public suffix division often provide multi-tenant services such as dynamic DNS, web hosting, and CDN services. As such, they sometimes allow their sub-tenants to provision names as subdomains of their public suffix. There are use-cases that require operators of domains in the public suffix list to demonstrate control over their domain, such as to be added to the Public Suffix List, or to provision a wildcard certificate. At the same time, if an operator of such a domain allows its customers or tenants to create names starting with an underscore ("_") then it opens up substantial risk to the domain operator for attackers to provision services on their domain.

Whether it is appropriate to allow domain verification on a public suffix will depend on the application. In the general case:

* Application Service Providers SHOULD NOT allow verification of ownership for domains which are public suffixes in the "ICANN" division. For example, "_example_service-challenge.co.uk" would not be allowed.

* Application Service Providers MAY allow verification of ownership for domains which are public suffixes in the "PRIVATE" division, although it would be preferable to apply additional safety checks in this case.

### 9.10. Unintentional Persistence

When persistent domain validation is used, a DNS Administrator failing to remove a no-longer desired Validation Record could enable a User to continue to have access to the domain within the Application Service Provider’s service. (H4 in Section 4.1)

When one-off domain validation is used, this is typically implemented through automation where a DNS Administrator grants the User access to make updates to the domain’s zone configuration. If the DNS Administrator fails to revoke access to a User who should no longer have access, this would enable the User to continue to perform new validations.

### 9.11. Reintroduction of Validation Records

When a domain has a new owner, that new owner could add a Validation Record that was present in the previous version of the domain. In the case of persistent validation this could be used to claim that the original User still has access to the domain within the Application Service Provider’s service. Applications implementing persistent domain validation need to include this risk within their threat model. (H1 and H4 in Section 4.1)

### 9.12. Amplification Attacks

Segmenting the Domain Control Validation tokens into individual per-service Validation Record Owner Names has the advantage of making the individual DNS responses smaller and thus reducing the potential of said TXT RRs to be used in the DNS amplification attacks. It should be noted that expired and no longer usable tokens should be removed even from Validation Record Owner Name DNS tree nodes to keep the DNS responses sizes at minimal level.

### 9.13. Validations not Coupled to Users

If an Application Service Provider does not properly associate Domain Validation with Users, the new owner of a domain could potentially gain access to Application Service Provider resources associated with the previous owner of a domain. Application Service Providers need to take care that re-validation of a domain by a different User is not necessarily treated as "reactivation" in a way that grants access to potentially sensitive resources stored and associated with a domain. (H2 in Section 4.1)

## 10. Privacy Considerations

As records are visible in the DNS they should be considered to be public information. While information in the Unique Token can be helpful to DNS Administrators, some constructions of Unique Tokens can leak information identifying a User either directly (e.g. containing the User’s identity or account identifier) or indirectly (e.g., an unkeyed hash of a username).

## 11. IANA Considerations

This document has no IANA actions.

## 12. References

### 12.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/rfc/rfc1034>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/rfc/rfc3339>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <https://www.rfc-editor.org/rfc/rfc4035>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC9364] Hoffman, P., "DNS Security Extensions (DNSSEC)", BCP 237, RFC 9364, DOI 10.17487/RFC9364, February 2023, <https://www.rfc-editor.org/rfc/rfc9364>.

## 12.2. Informative References

[ACME-DNS-ACCOUNT-LABEL]
Chariton, A., Omidi, A., Kasten, J., Loukos, F., and S. A. Janikowski, "Automated Certificate Management Environment (ACME) DNS Labeled With ACME Account ID Challenge", 2025, <https://datatracker.ietf.org/doc/draft-ietf-acme-dns-account-label/>.

[I-D.draft-tjw-dbound2-problem-statement]
Wicinski, T., "Domain Boundaries 2.0 Problem Statement", Work in Progress, Internet-Draft, draft-tjw-dbound2-problem-statement-01, 10 July 2023, <https://datatracker.ietf.org/doc/html/draft-tjw-dbound2-problem-statement-01>.

[PSL] Mozilla Foundation, "Public Suffix List", 2022, <https://publicsuffix.org/>.

[PSL-DIVISIONS]
Frakes, J., "Public Suffix List format", 2022, <https://github.com/publicsuffix/list/wiki/Format#divisions>.

[RFC3833] Atkins, D. and R. Austein, "Threat Analysis of the Domain Name System (DNS)", RFC 3833, DOI 10.17487/RFC3833, August 2004, <https://www.rfc-editor.org/rfc/rfc3833>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC4343] Eastlake 3rd, D., "Domain Name System (DNS) Case Insensitivity Clarification", RFC 4343, DOI 10.17487/RFC4343, January 2006, <https://www.rfc-editor.org/rfc/rfc4343>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/rfc/rfc5234>.

[RFC6672] Rose, S. and W. Wijngaards, "DNAME Redirection in the DNS", RFC 6672, DOI 10.17487/RFC6672, June 2012, <https://www.rfc-editor.org/rfc/rfc6672>.

[RFC7132] Kent, S. and A. Chi, "Threat Model for BGP Path Security", RFC 7132, DOI 10.17487/RFC7132, February 2014, <https://www.rfc-editor.org/rfc/rfc7132>.

[RFC8555] Barnes, R., Hoffman-Andrews, J., McCarney, D., and J. Kasten, "Automatic Certificate Management Environment (ACME)", RFC 8555, DOI 10.17487/RFC8555, March 2019, <https://www.rfc-editor.org/rfc/rfc8555>.

[RFC9210] Kristoff, J. and D. Wessels, "DNS Transport over TCP - Operational Requirements", BCP 235, RFC 9210, DOI 10.17487/RFC9210, March 2022, <https://www.rfc-editor.org/rfc/rfc9210>.

[RFC9499] Hoffman, P. and K. Fujiwara, "DNS Terminology", BCP 219, RFC 9499, DOI 10.17487/RFC9499, March 2024, <https://www.rfc-editor.org/rfc/rfc9499>.

[RFC9715] Fujiwara, K. and P. Vixie, "IP Fragmentation Avoidance in DNS over UDP", RFC 9715, DOI 10.17487/RFC9715, January 2025, <https://www.rfc-editor.org/rfc/rfc9715>.

[SUBDOMAIN-TAKEOVER] Mozilla, "Subdomain takeovers", n.d., <https://developer.mozilla.org/en-US/docs/Web/Security/Subdomain_takeovers>.

[UNDERSCORE-REGISTRY] IANA, "Underscored and Globally Scoped DNS Node Name", n.d., <https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#underscored-globally-scoped-dns-node-names>.

# Appendix A. Appendix

## A.1. Common Pitfalls

A very common but unfortunate technique in use today is to employ a DNS TXT record and placing it at the exact domain name whose control is being validated (e.g., often the zone apex). This has a number of known operational issues. If the User has multiple application services employing this technique, it will end up with multiple DNS TXT records having the same owner name; one record for each of the services.

Since DNS resource record sets are treated atomically, a query for the Validation Record will return all TXT records in the response. There is no way for the verifier to specifically query only the TXT record that is pertinent to their application service. The verifier must obtain the aggregate response and search through it to find the specific record it is interested in.

Additionally, placing many such TXT records at the same name increases the size of the DNS response. If the size of the UDP response (UDP being the most common DNS transport today) is large enough that it does not fit into the Path MTU of the network path, this may result in IP fragmentation, which can be unreliable due to firewalls and middleboxes is vulnerable to various attacks ([RFC9715]). Depending on message size limits configured or being negotiated, it may alternatively cause the DNS server to "truncate" the UDP response and force the DNS client to re-try the query over TCP in order to get the full response. Not all networks properly transport DNS over TCP and some DNS software mistakenly believe TCP support is optional ([RFC9210]). Huge TXT RRsets (due to many TXT records at the same name) can also be leveraged by attackers for traffic amplication attacks.

Other possible issues may occur. If a TXT record (or any other record type) is designed to be placed at the same domain name that is being validated, it may not be possible to do so if that name already has a CNAME record. This is because CNAME records cannot co-exist with other (non-DNSSEC) records at the same name. This situation cannot occur at the apex of a DNS zone, but can at a name deeper within the zone.

When multiple distinct services specify placing Validation Records at the same owner name, there is no way to delegate an application specific domain Validation Record to a third party. Furthermore, even without delegation, an organization may have a shared DNS zone where they need to provide record level permissions to the specific division within the organization that is responsible for the application in question. This can’t be done if all applications expect to find validation records at the same name.

### A.2. Domain Boundaries

The hierarchical structure of domain names do not necessarily define boundaries of ownership and administrative control (e.g., as discussed in [I-D.draft-tjw-dbound2-problem-statement]). Some domain names are "public suffixes" ([RFC9499]) where care may need to be taken when validating control. For example, there are security risks if an Application Service Provider can be tricked into believing that an attacker has control over ".co.uk" or ".com". The volunteer-managed Public Suffix List [PSL] is one mechanism available today that can be useful for identifying public suffixes.

Future specifications may provide better mechanisms or recommendations for defining domain boundaries or for enabling organizational administrators to place constraints on domains and subdomains.

### A.3. Interactions with DNAME

Domain control validation in the presence of a DNAME [RFC6672] is possible with caveats. Since a DNAME record redirects the entire subtree of names underneath the owner of the DNAME, it is not possible to place a Validation Record under the DNAME owner itself. It would have to be placed under the DNAME target name, since any lookups for a name under the DNAME owner will be redirected to the corresponding name under the DNAME target.

## Appendix B. Acknowledgments

Thank you to John Levine, Daniel Kahn Gillmor, Amir Omidi, Tuomo Soini, Ben Kaduk, Paul Hoffman, テ］gel Gonzテ。lez, Ondナ册j Surテス, and many others for their feedback and suggestions on this document.

## Authors’ Addresses

Shivan Sahib
Brave Software
Email: shivankaulsahib@gmail.com

Shumon Huque
Salesforce
Email: shuque@gmail.com

Paul Wouters
Aiven
Email: paul.wouters@aiven.io

Erik Nygren
Akamai Technologies
Email: erik+ietf@nygren.org

Tim Wicinski
Cox Communications
Email: tjw.ietf@gmail.com