Stream: Internet Engineering Task Force (IETF)
RFC: 9799
Category: Standards Track
Published: June 2025
ISSN: 2070-1721
Author: Q. Misell, Ed.
AS207960

RFC 9799
Automated Certificate Management Environment (ACME) Extensions for ".onion" Special-Use Domain Names

Abstract

This document defines extensions to the Automated Certificate Management Environment (ACME) to allow for the automatic issuance of certificates to Tor hidden services (".onion" Special-Use Domain Names).

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9799.

Table of Contents

1. Introduction

The Tor network has the ability to host "Onion Services" [tor-spec] only accessible via the Tor network. These services use the ".onion" Special-Use Domain Name [RFC7686] to identify these services. These can be used as any other domain name could, but they do not form part of the DNS infrastructure.

The Automated Certificate Management Environment (ACME) [RFC8555] defines challenges for validating control of DNS identifiers, and whilst a ".onion" Special-Use Domain Name may appear as a DNS name, it requires special consideration to validate control of one such that ACME could be used on ".onion" Special-Use Domain Names.

In order to allow ACME to be utilized to issue certificates to ".onion" Special-Use Domain Names, this document specifies challenges suitable to validate control of these Special-Use Domain Names. Additionally, this document defines an alternative to the DNS Certification Authority Authorization (CAA) Resource Record [RFC8659] that can be used with ".onion" Special-Use Domain Names.

1.1. Requirements Language

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.

2. Identifier

[RFC8555] defines the "dns" identifier type. This identifier type **MUST** be used when requesting a certificate for a ".onion" Special-Use Domain Name. The value of the identifier **MUST** be the textual representation as defined in the "Special Hostnames in Tor - .onion" section of [tor-spec]. The value **MAY** include subdomain labels. Version 2 addresses [tor-rend-spec-v2] **MUST NOT** be used as these are now considered insecure.

Example identifiers (line breaks have been added for readability only):

```
{
  "type": "dns",
  "value": "bbcweb3hytmzhn5d532owbu6oqadra5z3ar726v
        q5kgwwn6aucdccrad.onion"
}
```

```
{
  "type": "dns",
  "value": "www.bbcweb3hytmzhn5d532owbu6oqadra5z3ar726v
        q5kgwwn6aucdccrad.onion"
}
```

3. Identifier Validation Challenges

The CA/Browser Forum Baseline Requirements define methods accepted by the CA industry for validation of ".onion" Special-Use Domain Names (see Appendix B.2 of [cabf-br]). This document incorporates these methods into ACME challenges.

3.1. Existing Challenges

3.1.1. Existing: "dns-01" Challenge

The existing "dns-01" challenge **MUST NOT** be used to validate ".onion" Special-Use Domain Names as these domains are not part of the DNS.

3.1.2. Existing: "http-01" Challenge

The "http-01" challenge, as defined in Section 8.3 of [RFC8555], **MAY** be used to validate a ".onion" Special-Use Domain Name with the modifications defined in this document, namely those described in Sections 4 and 6.

The ACME server **SHOULD** follow redirects. Note that these **MAY** be redirects to services that are not ".onion" and that the server **SHOULD** honor these. For example, clients might use redirects so that the response can be provided by a centralized certificate management server. See Section 10.2 of [RFC8555] for security considerations on why a server might not want to follow redirects.

3.1.3. Existing "tls-alpn-01" Challenge

The "tls-alpn-01" challenge, as defined in [RFC8737], **MAY** be used to validate a ".onion" Special-Use Domain Name with the modifications defined in this document, namely those described in Sections 4 and 6.

3.2. New "onion-csr-01" Challenge

The two ACME-defined methods allowed by CA/BF described in Sections 3.1.2 and 3.1.3 ("http-01" and "tls-alpn-01") do not allow issuance of wildcard certificates. A ".onion" Special-Use Domain Name can have subdomains (just like any other domain in the DNS), and a site operator may find it useful to have one certificate for all virtual hosts on their site. This new validation method incorporates the specially signed Certificate Signing Request (CSR) (as defined by Appendix B.2.b of [cabf-br]) into ACME to allow for the issuance of wildcard certificates.

To this end, a new challenge called "onion-csr-01" is defined, with the following fields:

type (required, string): The string `"onion-csr-01"`.

nonce (required, string): A Base64-encoded nonce [RFC4648] including padding characters. It **MUST** contain at least 64 bits of entropy. A response generated using this nonce **MUST NOT** be accepted by the ACME server if the nonce used was generated by the server more than 30 days prior (as per Appendix B.2.b of [cabf-br]).

authKey (optional, object): The ACME server's Ed25519 public key encoded as per [RFC8037]. This is further defined in Section 4.

```
{
  "type": "onion-csr-01",
  "url": "https://acme-server.example.onion/acme/chall/bbc625c5",
  "status": "pending",
  "nonce": "bI6/MRqV4gw=",
  "authKey": { ... }
}
```

An "onion-csr-01" challenge **MUST NOT** be used to issue certificates for Special-Use Domain Names that are not ".onion".

Clients prove control over the key associated with the ".onion" service by generating a CSR [RFC2986] with the following additional extension attributes and signing it with the private key of the ".onion" Special-Use Domain Name:

- A `caSigningNonce` attribute containing the nonce provided in the challenge. This **MUST** be raw bytes and not the base64 encoded value provided in the challenge object.
- An `applicantSigningNonce` attribute containing a nonce generated by the client. This **MUST** have at least 64 bits of entropy. This **MUST** be raw bytes.

These additional attributes have the following format

```
cabf OBJECT IDENTIFIER ::=
  { joint-iso-itu-t(2) international-organizations(23)
    ca-browser-forum(140) }

cabf-caSigningNonce OBJECT IDENTIFIER ::= { cabf 41 }

caSigningNonce ATTRIBUTE ::= {
  WITH SYNTAX             OCTET STRING
  EQUALITY MATCHING RULE  octetStringMatch
  SINGLE VALUE            TRUE
  ID                      { cabf-caSigningNonce }
}

cabf-applicantSigningNonce OBJECT IDENTIFIER ::= { cabf 42 }

applicantSigningNonce ATTRIBUTE ::= {
  WITH SYNTAX             OCTET STRING
  EQUALITY MATCHING RULE  octetStringMatch
  SINGLE VALUE            TRUE
  ID                      { cabf-applicantSigningNonce }
}
```

The subject of the CSR need not be meaningful and CAs **MUST NOT** validate its contents. The public key presented in this CSR **MUST** be the public key corresponding to the ".onion" Special-Use Domain Name being validated. It **MUST NOT** be the same public key presented in the CSR to finalize the order.

Clients respond with the following object to validate the challenge:

csr (required, string): The CSR in the base64url-encoded version of the DER format. (Note: Because this field uses base64url, and does not include headers, it is different from Privacy Enhanced Mail (PEM).)

```
POST /acme/chall/bbc625c5
Host: acme-server.example.onion
Content-Type: application/jose+json

{
  "protected": base64url({
    "alg": "ES256",
    "kid": "https://acme-server.example.onion/acme/acct/evOfKhNU60wg",
    "nonce": "UQI1PoRi5OuXzxuX7V7wL0",
    "url": "https://acme-server.example.onion/acme/chall/bbc625c5"
  }),
  "payload": base64url({
    "csr": "MIIBPTCBxAIBADBFMQ...FS6aKdZeGsysoCo4H9P"
  }),
  "signature": "Q1bURgJoEslbD1c5...3pYdSMLio57mQNN4"
}
```

When presented with the CSR, the server verifies it in the following manner:

1. The CSR is a well formatted PKCS#10 request.
2. The public key in the CSR corresponds to the ".onion" Special-Use Domain Name being validated.
3. The signature over the CSR validates with the ".onion" Special-Use Domain Name public key.
4. The caSigningNonce attribute is present and its contents match the nonce provided to the client.
5. The applicantSigningNonce attribute is present and contains at least 64 bits of entropy.

If all of the above are successful then validation succeeds, otherwise it has failed.

4. Client Authentication to Hidden Services

Some hidden services do not wish to be accessible to the entire Tor network, and so they encrypt their hidden service descriptor with the keys of clients authorized to connect. Without a way for the CA to signal what key it will use to connect, these services will not be able to obtain a certificate using http-01 or tls-alpn-01, nor enforce CAA with any validation method.

To this end, an additional field in the challenge object is defined to allow the ACME server to advertise the Ed25519 public key it will use (as per the "Authentication during the introduction phase" section of [tor-spec]) to authenticate itself when retrieving the hidden service descriptor.

authKey (optional, object): The ACME server's Ed25519 public key encoded as per [RFC8037].

ACME servers **MUST NOT** use the same public key with multiple hidden services. ACME servers **MAY** reuse public keys for re-validation of the same hidden service.

There is no method to communicate to the CA that client authentication is necessary; instead, the ACME server **MUST** attempt to calculate its CLIENT-ID as per the "Client behavior" section of [tor-spec]. If no `auth-client` line in the first layer hidden service descriptor matches the computed client-id, then the server **MUST** assume that the hidden service does not require client authentication and proceed accordingly.

In the case in which the Ed25519 public key is novel to the client, it will have to resign and republish its hidden service descriptor. It **MUST** wait some (indeterminate) amount of time for the new descriptor to propagate the Tor hidden service directory servers before proceeding with responding to the challenge. This should take no more than a few minutes. This specification does not set a fixed time as changes in the operation of the Tor network can affect this propagation time in the future. ACME servers **MUST NOT** expire challenges before a reasonable time to allow publication of the new descriptor. It is **RECOMMENDED** the server allow at least 30 minutes; however, it is entirely up to operator preference.

5. ACME over Hidden Services

A CA offering certificates to ".onion" Special-Use Domain Names **SHOULD** make their ACME server available as a Tor hidden service. ACME clients **SHOULD** also support connecting to ACME servers over Tor, regardless of their support of "onion-csr-01", as their existing "http-01" and "tls-alpn-01" implementations could be used to obtain certificates for ".onion" Special-Use Domain Names.

6. Certification Authority Authorization (CAA)

".onion" Special-Use Domain Names are not part of the DNS; as such, a variation on CAA [RFC8659] is necessary to allow restrictions to be placed on certificate issuance.

To this end, a new field is added to the second layer hidden service descriptor, as defined in the "Second layer plaintext format" section of [tor-spec] with the following format (defined using the notation from the "netdoc document meta-format" section of [tor-spec]):

```
"caa" SP flags SP tag SP value NL
[Any number of times]
```

The presentation format is provided above purely for the convenience of the reader and implementors: the canonical version remains that defined in Section 4.1.1 of [RFC8659], or future updates to the same.

The contents of "flags", "tag", and "value" are as per Section 4.1.1 of [RFC8659]. Multiple CAA records **MAY** be present, as is the case in the DNS. CAA records in a hidden service descriptor are to be treated the same by CAs as if they had been in the DNS for the ".onion" Special-Use Domain Name.

A hidden service's second layer descriptor using CAA could look something like the following (additional line breaks have been added for readability):

```
create2-formats 2
single-onion-service
caa 128 issue "acmeforonions.example;validationmethods=onion-csr-01"
caa 0 iodef "mailto:security@example.com"
introduction-point AwAGsAk5nSMpAhRqhMHbTFCTSlfhP8f5PqUhe6DatgMgk7kSL3
        KHCZUZ3C6tXDeRfM9SyNY0DlgbF8q+QSaGKCs=
...
```

6.1. Relevant Resource Record Set

In the absence of the possibility for delegation of subdomains from a ".onion" Special-Use Domain Name, as there is in the DNS, there is no need, nor indeed any method available, to search up the DNS tree for a relevant CAA record set. Similarly, it is also impossible to check CAA records on the "onion" Special-Use Top-Level Domain (TLD), as it does not exist in any form except as described in [RFC7686]; therefore, implementors **MUST NOT** look there either.

Instead, all subdomains under a ".onion" Special-Use Domain Name share the same CAA record set. That is, all of these share a CAA record set with "a.onion":

- b.a.onion
- c.a.onion
- e.d.a.onion

but these do not:

- b.c.onion
- c.d.onion
- e.c.d.onion
- a.b.onion

6.2. When to Check CAA

If the hidden service has client authentication enabled, then it will be impossible for the ACME server to decrypt the second layer descriptor to read the CAA records until the ACME server's public key has been added to the first layer descriptor. To this end, an ACME server **MUST** wait until the client responds to an authorization before checking the CAA and treat this response as an indication that their public key has been added and that the ACME server will be able to decrypt the second layer descriptor.

6.3. Preventing Mis-Issuance by Unknown CAs

In the case of a hidden service requiring client authentication, the CA will be unable to read the hidden service's CAA records without the hidden service trusting an ACME server's public key -- as the CAA records are in the second layer descriptor. A method is necessary to signal that there are CAA records present (but not reveal their contents, which, in certain circumstances, would unwantedly disclose information about the hidden service operator).

To this end, a new field is added to the first layer hidden service descriptor in the "First layer plaintext format" section of [tor-spec] with the following format (defined using the notation from the "netdoc document meta-format" section of [tor-spec]):

```
"caa-critical" NL
[At most once]
```

If an ACME server encounters this flag, it **MUST NOT** proceed with issuance until it can decrypt and parse the CAA records from the second layer descriptor.

6.4. Alternative In-Band Presentation of CAA

An ACME server might be unwilling to operate the infrastructure required to fetch, decode, and verify Tor hidden service descriptors in order to check CAA records. To this end a method to signal CAA policies in-band of ACME is defined.

If a hidden service does use this method to provide CAA records to an ACME server, it **SHOULD** still publish CAA records if its CAA record set includes "iodef", "contactemail", or "contactphone" so that this information is still publicly accessible. A hidden service operator **MAY** also not wish to publish a CAA record set in its service descriptor to avoid revealing information about the service operator.

If an ACME server receives a validly signed CAA record set in the finalize request, it **MAY** proceed with issuance on the basis of the client-provided CAA record set only, without checking the CAA set in the hidden service. Alternatively, an ACME server **MAY** ignore the client provided record set and fetch the record set from the service descriptor. In any case, the server always **MAY** fetch the record set from the service descriptor. If an ACME server receives a validly signed CAA record set in the finalize request, it need not check the CAA set in the hidden service descriptor and can proceed with issuance on the basis of the client-provided CAA record set only. An ACME server **MAY** ignore the client-provided record set and is free to always fetch the record set from the service descriptor.

A new field is defined in the ACME finalize endpoint to contain the hidden service's CAA record set for each ".onion" Special-Use Domain Name in the order.

onionCAA (optional, dictionary of objects): The CAA record set for each ".onion" Special-Use Domain Name in the order. The key is the ".onion" Special-Use Domain Name, and the value is an object with the fields described below.

The contents of the values of the "onionCAA" object are as follows:

caa (required, string or null): The CAA record set as a string, encoded in the same way as if was included in the hidden service descriptor. If the hidden service does not have a CAA record set, then this **MUST** be null.

expiry (required, integer): The Unix timestamp at which this CAA record set will expire. This **SHOULD NOT** be more than 8 hours in the future. ACME servers **MUST** process this as at least a 64-bit integer to ensure functionality beyond 2038.

signature (required, string): The Ed25519 signature of the CAA record set using the private key corresponding to the ".onion" Special-Use Domain Name, encoded using base64url. The signature is defined below.

The data that the signature is calculated over is the concatenation of the following, encoded in UTF-8 [RFC3629]:

```
"onion-caa|" || expiry || "|" || caa
```

Where "|" is the ASCII character 0x7C, and expiry is the expiry field as a decimal string with no leading zeros. If the caa field is null, it is represented as an empty string in the signature calculation.

6.4.1. ACME Servers Requiring In-Band CAA

If an ACME server does not support fetching a service's CAA record set from its service descriptor, and the ACME client does not provide an "onionCAA" object in its finalize request, the ACME server **MUST** respond with an "onionCAARequired" error to indicate this.

To support signaling the server's support for fetching CAA record sets over Tor, a new field is defined in the directory "meta" object to signal this.

inBandOnionCAARequired (optional, boolean): If true, the ACME server requires the client to provide the CAA record set in the finalize request. If false or absent, the ACME server does not require the client to provide the CAA record set is this manner.

A directory of such a CA could look like the following:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "newNonce": "https://acme-server.example.onion/acme/new-nonce",
  "newAccount": "https://acme-server.example.onion/acme/new-account",
  "newOrder": "https://acme-server.example.onion/acme/new-order",
  "revokeCert": "https://acme-server.example.onion/acme/revoke-cert",
  "keyChange": "https://acme-server.example.onion/acme/key-change",
  "meta": {
    "termsOfService": "https://acme-server.example.onion/acme/terms/
2023-10-13",
    "website": "https://acmeforonions.example/",
    "caaIdentities": ["acmeforonions.example"],
    "inBandOnionCAARequired": true
  }
}
```

6.4.2. Example In-Band CAA

Given the following example CAA record set for 5anebu2glyc235wbbop3m2ukzlaptpkq333vdtdvcjpigyb7x2i2m2qd.onion (additional line breaks have been added for readability):

```
caa 128 issue "acmeforonions.example;
           validationmethods=onion-csr-01"
caa 0 iodef "mailto:example@example.com"
```

The following would be submitted to the ACME server's finalize endpoint (additional line breaks have been added for readability):

```
POST /acme/order/TOlocE8rfgo/finalize
Host: acme-server.example.onion
Content-Type: application/jose+json

{
  "protected": base64url({
    "alg": "ES256",
    "kid": "https://acme-server.example.onion/acme/acct/evOfKhNU60wg",
    "nonce": "MSF2j2nawWHPxxkE3ZJtKQ",
    "url": "https://acme-server.example.onion/acme/order/TOlocE8rfgo/
finalize"
  }),
  "payload": base64url({
    "csr": "MIIBPTCBxAIBADBFMQ...FS6aKdZeGsysoCo4H9P",
    "onionCAA": {
      "5anebu2glyc235wbbop3m2ukzlaptpkq333vdtdvcjpi
           gyb7x2i2m2qd.onion": {
        "caa": "caa 128 issue \"acmeforonions.example;
           validationmethods=onion-csr-01\"\n
           caa 0 iodef \"mailto:example@example.com\"",
        "expiry": 1697210719,
        "signature": "u_iP6JZ4JZBrzQUKH6lSrWejjRfeQmkTuehc0_FaaTNP
           AV0RVxpUz9r44DRdy6kgy0ofnx18KIhMrP7N1wpxAA=="
      }
    }
  }),
  "signature": "uOrUfIIk5RyQ...nw62Ay1cl6AB"
}
```

7. IANA Considerations

7.1. Validation Methods

One new entry has been added to the "ACME Validation Methods" registry that was defined in Section 9.7.8 of [RFC8555] (<https://www.iana.org/assignments/acme>).

Label	Identifier Type	ACME	Reference
onion-csr-01	dns	Y	This document

Table 1: onion-csr-01 Validation Method

7.2. Error Types

One new entry has been added to the "ACME Error Types" registry that was defined in Section 9.7.4 of [RFC8555] (<https://www.iana.org/assignments/acme>).

Type	Description	Reference
onionCAARequired	The CA only supports checking the CAA for hidden services in-band, but the client has not provided an in-band CAA	This document

Table 2: onionCAARequired Error Type

7.3. Directory Metadata Fields

One new entry has been added to the "ACME Directory Metadata Fields" registry that was defined in Section 9.7.6 of [RFC8555] (<https://www.iana.org/assignments/acme>).

Field name	Field type	Reference
onionCAARequired	boolean	This document

Table 3: onionCAARequired Metadata Field

8. Security Considerations

8.1. Security of the "onion-csr-01" Challenge

The security considerations of [cabf-br] apply to issuance using the CSR method.

8.2. Use of the "dns" Identifier Type

The reuse of the "dns" identifier type for a Special-Use Domain Name not actually in the DNS infrastructure raises questions regarding its suitability. The reasons to pursue this path in the first place are detailed in Appendix A. It is felt that there is little security concern in reuse of the "dns" identifier type with regard to the mis-issuance by CAs that are not aware of ".onion" Special-Use Domain Names as CAs would not be able to resolve the identifier in the DNS.

8.2.1. "http-01" Challenge

In the absence of knowledge of this document, a CA would follow the procedure set out in Section 8.3 of [RFC8555], which specifies that the CA should "Dereference the URL using an HTTP GET request". Given that ".onion" Special-Use Domain Names require special handling to dereference, this dereferencing will fail, disallowing issuance.

8.2.2. "tls-alpn-01" Challenge

In the absence of knowledge of this document, a CA would follow the procedure set out in Section 3 of [RFC8737], which specifies that the CA "resolves the domain name being validated and chooses one of the IP addresses returned for validation". Given that ".onion" Special-Use Domain Names are not resolvable to IP addresses, this dereferencing will fail, disallowing issuance.

8.2.3. "dns-01" Challenge

In the absence of knowledge of this document, a CA would follow the procedure set out in Section 8.4 of [RFC8555], which specifies that the CA should "query for TXT records for the validation domain name". Given that ".onion" Special-Use Domain Names are not present in the DNS infrastructure, this query will fail, disallowing issuance.

8.3. Key Authorization with "onion-csr-01"

The "onion-csr-01" challenge does not make use of the key authorization string defined in Section 8.1 of [RFC8555]. This does not weaken the integrity of authorizations.

The key authorization exists to ensure that, whilst an attacker observing the validation channel can observe the correct validation response, they cannot compromise the integrity of authorizations as the response can only be used with the account key for which it was generated. As the validation channel for this challenge is ACME itself, and ACME already requires that the request be signed by the account, the key authorization is not necessary.

8.4. Use of Tor for Domains That Are Not ".onion"

An ACME server **MUST NOT** utilize Tor for the validation of domains that are not ".onion", due to the risk of exit hijacking [spoiled-onions].

8.5. Redirects with "http-01"

A site **MAY** redirect to another site when completing validation using the "http-01" challenge. This redirect **MAY** be to either another ".onion" Special-Use Domain Name or a domain in the public DNS. A site operator **MUST** consider the privacy implications of redirecting to a site that is not ".onion" -- namely that the ACME server operator will then be able to learn information about the site they were redirected to that they would not have if accessed via a ".onion" Special-Use Domain Name, such as its IP address. If the site redirected to is on the same or an adjacent host to the ".onion" Special-Use Domain Name, this reveals information that the "Tor Rendezvous Specification - Version 3" secion of [tor-spec] was otherwise designed to protect.

If an ACME server receives a redirect to a domain in the public DNS, it **MUST NOT** utilize Tor to make a connection to it due to the risk of exit hijacking.

8.6. Security of CAA Records

The second layer descriptor is signed, encrypted, and encoded using Message Authentication Code (MAC) in a way that only a party with access to the secret key of the hidden service could manipulate what is published there. For more information about this process, see the "Hidden service descriptors: encryption format" section of [tor-spec].

8.7. In-Band CAA

Tor directory servers are inherently untrusted entities; as such, there is no difference in the security model for accepting CAA records directly from the ACME client or fetching them over Tor. There is no difference in the security model between accepting CAA records directly from the ACME client and fetching them over Tor; the CAA records are verified using the same hidden service key in either case.

8.8. Access of the Tor Network

The ACME server **MUST** make its own connection to the hidden service via the Tor network and **MUST NOT** outsource this to a third-party service, such as Tor2Web.

8.9. Anonymity of the ACME Client

ACME clients requesting certificates for ".onion" Special-Use Domain Names not over the Tor network can inadvertently expose the existence of a hidden service on the host requesting certificates to unintended parties; this is true even when features such as Encrypted ClientHello (ECH) [tls-esni] are utilized, as the IP addresses of ACME servers are generally well-known, static, and not used for any other purpose.

ACME clients **SHOULD** connect to ACME servers over the Tor network to alleviate this, preferring a hidden service endpoint if the CA provides such a service.

If an ACME client requests a publicly trusted WebPKI certificate, it will expose the existence of the Hidden Service publicly due to its inclusion in Certificate Transparency logs [RFC9162]. Hidden Service operators **MUST** consider the privacy implications of this before requesting WebPKI certificates. ACME client developers **SHOULD** warn users about the risks of CT-logged certificates for hidden services.

8.9.1. Avoid Unnecessary Certificates

Not all services will need a publicly trusted WebPKI certificate; for internal or non-public services, operators **SHOULD** consider using self-signed or privately trusted certificates that aren't logged to certificate transparency.

8.9.2. Obfuscate Subscriber Information

When an ACME client is registering with an ACME server, it **SHOULD** provide minimal or obfuscated subscriber details to the CA, such as a pseudonymous email address, if at all possible.

8.9.3. Separate ACME Account Keys

If a hidden service operator does not want their different hidden services to be correlated by a CA, they **MUST** use separate ACME account keys for each hidden service.

9. References

9.1. Normative References

[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/info/rfc2119>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

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

[RFC7686] Appelbaum, J. and A. Muffett, "The ".onion" Special-Use Domain Name", RFC 7686, DOI 10.17487/RFC7686, October 2015, <https://www.rfc-editor.org/info/rfc7686>.

[RFC8037] Liusvaara, I., "CFRG Elliptic Curve Diffie-Hellman (ECDH) and Signatures in JSON Object Signing and Encryption (JOSE)", RFC 8037, DOI 10.17487/RFC8037, January 2017, <https://www.rfc-editor.org/info/rfc8037>.

[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/info/rfc8174>.

[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/info/rfc8555>.

[RFC8659] Hallam-Baker, P., Stradling, R., and J. Hoffman-Andrews, "DNS Certification Authority Authorization (CAA) Resource Record", RFC 8659, DOI 10.17487/RFC8659, November 2019, <https://www.rfc-editor.org/info/rfc8659>.

[RFC8737] Shoemaker, R.B., "Automated Certificate Management Environment (ACME) TLS Application-Layer Protocol Negotiation (ALPN) Challenge Extension", RFC 8737, DOI 10.17487/RFC8737, February 2020, <https://www.rfc-editor.org/info/rfc8737>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[tor-spec] The Tor Project, "Tor Specifications", <https://spec.torproject.org>.

[tor-rend-spec-v2] The Tor Project, "Tor Rendezvous Specification - Version 2", commit 2437d19c, <https://spec.torproject.org/rend-spec-v2>.

[cabf-br] CA/Browser Forum, "Baseline Requirements for the Issuance and Management of Publicly-Trusted TLS Server Certificates", Version 2.0.6, 5 August 2024, <https://cabforum.org/working-groups/server/baseline-requirements/documents/CA-Browser-Forum-TLS-BR-2.0.6.pdf>.

9.2. Informative References

[onion-services-setup] The Tor Project, "Set Up Your Onion Service", <https://community.torproject.org/onion-services/setup/>.

[spoiled-onions] Winter, P., Köwer, R., Mulazzani, M., Huber, M., Schrittwieser, S., Lindskog, S., and E. Weippl, "Spoiled Onions: Exposing Malicious Tor Exit Relays", Privacy Enhancing Technologies (PETS 2014), Lecture Notes in Computer Science, vol. 8555, pp. 304-331, DOI 10.1007/978-3-319-08506-7_16, 2014, <https://doi.org/10.1007/978-3-319-08506-7_16>.

[tls-esni] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-24, 20 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-24>.

[RFC9162] Laurie, B., Messeri, E., and R. Stradling, "Certificate Transparency Version 2.0", RFC 9162, DOI 10.17487/RFC9162, December 2021, <https://www.rfc-editor.org/info/rfc9162>.

Appendix A. Discussion on the Use of the "dns" Identifier Type

The reasons for utilizing the "dns" identifier type in ACME and not defining a new identifier type for ".onion" may not seem obvious at first glance. After all, ".onion" Special-Use Domain Names are not part of the DNS infrastructure and, as such, why should they use the "dns" identifier type?

Appendix B.2.a.ii of [cabf-br] defines, and this document allows, using the "http-01" or "tls-alpn-01" validation methods already present in ACME (with some considerations). Given the situation of a web server placed behind a Tor-terminating proxy (as per the setup suggested by the Tor project [onion-services-setup]), existing ACME tooling can be blind to the fact that a ".onion" Special-Use Domain Name is being utilized, as they simply receive an incoming TCP connection as they would regardless (albeit from the Tor-terminating proxy).

An example of this would be Certbot placing the ACME challenge response file in the webroot of an NGINX web server. Neither Certbot nor NGINX would require any modification to be aware of any special handling for ".onion" Special-Use Domain Names.

This does raise some questions regarding security within existing implementations; however, the authors believe this is of little concern, as per Section 8.2.

Acknowledgements

With thanks to the Open Technology Fund for funding the work that went into this document.

The authors also wish to thank the following for their input on this document:

- Iain Learmonth
- Jan-Frederik Rieckers

Author's Address

Q Misell (EDITOR)
AS207960 Cyfyngedig
13 Pen-y-lan Terrace
Caerdydd
CF23 9EU
United Kingdom
Email: q@as207960.net, q@magicalcodewit.ch
URI: https://magicalcodewit.ch