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

Abstract

The 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).

Discussion

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/AS207960/acme-onion.

The project website and a reference implementation can be found at https://acmeforonions.org.

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/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 18 April 2024.
Table of Contents

1. Introduction
   1.1. Requirements Language

2. Identifier

3. Identifier Validation Challenges
   3.1. Existing challenges
      3.1.1. Existing "dns-01" Challenge
      3.1.2. Existing "http-01" Challenge
      3.1.3. Existing "tls-alpn-01" Challenge
   3.2. New "onion-csr-01" Challenge

4. Client authentication to hidden services

5. ACME over hidden services

6. Certification Authority Authorization (CAA)
   6.1. Relevant Resource Record Set
   6.2. When to check CAA
   6.3. Preventing mis-issuance by unknown CAs
   6.4. Alternative in-band presentation of CAA
      6.4.1. CAs requiring in-band CAA
      6.4.2. Example in-band CAA

7. IANA Considerations
   7.1. Validation Methods
   7.2. Error Types
   7.3. Directory Metadata Fields

8. Security Considerations
   8.1. Use of "dns" identifier type
      8.1.1. "http-01" Challenge
      8.1.2. "tls-alpn-01" Challenge
      8.1.3. "dns-01" Challenge
   8.2. Key Authorization with "onion-csr-01"
   8.3. Use of Tor for non ".onion" domains
   8.4. Security of CAA records
   8.5. In-band CAA
   8.6. Access of the Tor network
1. Introduction

The Tor network has the ability to host "Onion Services" [tor-rend-spec-v3] [tor-address-spec] only accessible via the Tor network. These use the "onion" Special-Use Domain Name [RFC7686] to identify these services. These can be used as any other domain name could, but 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 utilised 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 [BCP14] 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 identifier MUST be the textual representation as defined in [tor-address-spec] §3. The value MAY include subdomain labels. Version 2 addresses MUST NOT be used as these are now considered insecure.

Example identifiers:
3. Identifier Validation Challenges

The CA/Browser Forum Baseline Requirements [cabf-br] §B.2 define methods accepted by the CA industry for validation of ".onion" Special-Use Domain Names. 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.

3.1.2. Existing "http-01" Challenge

The "http-01" challenge is defined as in [RFC8555] §8.3 may be used to validate a ".onion" Special-Use Domain Names, with the modifications defined in this standard.

The ACME server SHOULD follow redirects; note that these may be redirects to non ".onion" services, and the server SHOULD honour these.

3.1.3. Existing "tls-alpn-01" Challenge

The "tls-alpn-01" challenge is defined as in [RFC8737] may be used to validate a ".onion" Special-Use Domain Names, with the modifications defined in this standard.

3.2. New "onion-csr-01" Challenge

The two methods already defined in ACME and allowed by the CA/BF do not allow issuance of wildcard certificates. This new validation method incorporates the specially signed CSR (as defined by [cabf-br] § B.2.b) into ACME to allow for the issuance of wildcard certificates.

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

```json
{
  "type": "dns",
  "value": "bbcweb3hytmzhn5d532owbu6oqadra5z3ar726vq5kgwbn6aucdccrad.onion"
}
{
  "type": "dns",
  "value": "www.bbcweb3hytmzhn5d532owbu6oqadra5z3ar726vq5kgwbn6aucdccrad"
}
```
type (required, string)  
The string "onion-csr-01"

nonce (required, string)  
A Base64 \[RFC4648\] encoded nonce,  
including padding characters. It MUST contain at least 64 bits of  
entropy. It MUST NOT be valid for more than 30 days.

authKey (optional, object)  
The Ed25519 public key encoded as per  
\[RFC8037\].

{/}

Clients prove control over the key associated with the ".onion"  
service by generating a CSR 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 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
The subject of the CSR need not be meaningful and CAs **SHOULD 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.

Client 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 PEM.)

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

{
    "protected": base64url({
        "alg": "ES256",
        "kid": "https://example.com/acme/acct/evOfKhNU60wg",
        "nonce": "UQI1PoRi5OuXzxuX7V7wL0",
        "url": "https://example.com/acme/chall/bbc625c5"
    }),
    "payload": base64url({
        "csr": "MIIBPTCBxAIBADBFMQ...FS6aKdZeGsysoCo4H9P"
    }),
    "signature": "Q1bURgJoEs1bD1c5...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 matches 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 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 [tor-rend-spec-v3] INTRO-AUTH) to authenticate itself when retrieving the hidden service descriptor.

`authKey (optional, object)` The Ed25519 public key encoded as per [RFC8037].

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

There is no method to communicate to the CA that client authentication is required; instead the ACME server **MUST** attempt to calculate its CLIENT-ID as per [tor-rend-spec-v3] FIRST-LAYER-CLIENT-BEHAVIOR. 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 the Ed25519 public key is novel to the client it will have to resign and republish its hidden service descriptor. It **SHOULD** 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. CAs **MUST NOT** expire challenges before a reasonable time to allow publication of the new descriptor (this document suggests at least 30 minutes).

5. ACME over hidden services

A CA offering certificates to ".onion" Special-Use Domain Names **SHOULD** strongly consider making their ACME server available as a Tor hidden services. 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 Name are not part of the DNS, and as such a variation on CAA [RFC8659] is required to allow restrictions to be placed on certificate issuance.

To this end a new field is added to the second layer hidden service descriptor [tor-rend-spec-v3] § 2.5.2.2. with the following format:

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

The contents of "flag", "tag", and "value" are as per [RFC8659] § 4.1.1. 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 may look something like the following:

```
create2-formats 2
single-onion-service
caa 128 issue "test.acmeforonions.org;validationmethods=onion-csr-01"
caa 0 iodef "mailto:security@example.com"
introduction-point AwAGsAk5nSMpAhRqhMHbTFCTSIfhP8f5PqUhe6DatgMgk7kSL3KHC.
...
```

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 TLD, as it does not exist in any form except as described in [RFC7686], so implementors must not look here 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

6.2. When to check CAA

If the hidden service has client authentication enabled then it will be impossible for the CAA to decrypt the second layer descriptor to read the CAA records until the CAAs public key has been added to first layer descriptor. To this end a CA SHOULD wait until the client responds to an authorization, and treat this as indication that their public key has been added and that the CA will be able to decrypt the second layer descriptor.

6.3. Preventing mis-issuance by unknown CAs

As the CAA records are in the second layer descriptor and in the case of a hidden service requiring client authentication it is impossible to read them without the hidden service trusting a CA's public key, a method is required to signal that there are CAA records present (but not reveal their contents, which may disclose unwanted information about the hidden service operator).

To this end a new field is added to the first layer hidden service descriptor [tor-rend-spec-v3] § 2.5.1.2. with the following format:

"caa-critical" NL
[At most once]
If a CA 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

A CA may not be willing 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 a CA 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 a CA 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. A CA, however, is not required to do anything with 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 following fields.

The contents of the "onionCAA" object are:

**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. CAs **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.

6.4.1. CAs requiring in-band CAA

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

Additionally, a new field is defined in the directory "meta" object to signal this.

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

A directory of such a CA may look like

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

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

6.4.2. Example in-band CAA

Given the following example CAA record set for 5anebu2g1yc235wbbop3m2ukz1aptpkq333vtdvcjpiyg7x2i2m2qd.onion:
The following would be submitted to the CA's finalize endpoint

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

{
    "protected": base64url({
        "alg": "ES256",
        "kid": "https://example.com/acme/acct/evOfKhNU60wg",
        "nonce": "MSF2j2nawWHPxxkE3ZJtKQ",
        "url": "https://example.com/acme/order/TOlocE8rfgo/finalize"
    }),
    "payload": base64url({
        "csr": "MIIBPTCBxAIBADBFMQ...FS6aKdZeGsysoCo4H9P",
        "onionCAA": {
            "5anebu2glyc235wbbop3m2ukzlaptpkq333vdtvcjpiygb7xzi2m2qd.onion": {
                "caa": "caa 128 issue "test.acmeforonions.org; validationmethods=onion-csr-01"
            }
        },
        "expiry": 1697210719,
        "signature": "u_iP6JZ4JZBrzQUKH61SrWejjRfeQmkTuehc0_FaaTNPAV0RVx}
    }),
    "signature": "uOrUfIIk5RyQ...nw62Ay1cl6AB"
}

7. IANA Considerations

7.1. Validation Methods

Per this document, one new entry has been added to the "ACME Validation Methods" registry defined in [RFC8555] §9.7.8. This entry is defined below:

<table>
<thead>
<tr>
<th>Label</th>
<th>Identifier Type</th>
<th>ACME</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>onion-csr-01</td>
<td>dns</td>
<td>Y</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 1: New entries

7.2. Error Types

Per this document, one new entry has been added to the "ACME Error Types" registry defined in [RFC8555] §9.7.8. This entry is defined below:
7.3. Directory Metadata Fields

Per this document, one new entry has been added to the "ACME Directory Metadata Fields" registry defined in [RFC8555] §9.7.8. This entry is defined below:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Field type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>onionCAARquired</td>
<td>boolean</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 2: New entries

8. Security Considerations

8.1. Use of "dns" identifier type

The re-use of the "dns" identifier type for a Special-Use Domain Name not actually in the DNS infrastructure raises questions regarding its suitability. The reasons the author wishes 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 regards 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.

8.1.1. "http-01" Challenge

The CA would follow the procedure set out in [RFC8555] §8.3 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 de-referencing will fail, disallowing issuance.

8.1.2. "tls-alpn-01" Challenge

The CA would follow the procedure set out in [RFC8737] §3 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 de-referencing will fail, disallowing issuance.
8.1.3. "dns-01" Challenge

The CA would follow the procedure set out in [RFC8555] §8.4 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.

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

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

The key authorization exists to ensure that whilst an attacker observing the validation channel may 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 required.

8.3. Use of Tor for non ".onion" domains

An ACME server **MUST NOT** utilise Tor for the validation of non ".onion" domains, due to the risk of possible exit hijacking.

8.4. Security of CAA records

The second layer descriptor is signed, encrypted and MACed 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 [tor-rend-spec-v3] § 2.5.3.

8.5. In-band CAA

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

8.6. 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, such as by using Tor2Web.
8.7. Anonymity of the ACME client

ACME clients requesting certificates for ".onion" Special-Use Domain Names may expose the existence of a hidden service on the host to unintended parties - even when features such as ECH [I-D.ietf-tls-esni] are utilised, 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.

9. References

9.1. Normative References

<https://www.rfc-editor.org/info/bcp14>


9.2. Informative References


Appendix A. Discussion on the use of the "dns" identifier type

The reasons for utilising the "dns" identifier type in ACME and not defining a new identifier type for ".onion" s 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?

The CA/Browser Forum Baseline Requirements [cabf-br] §B.2.a.ii define, and this standard 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 utilised, 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 8.1.

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 R. Learmonth

*Jan-Frederik Rieckers

Author's Address

Q Misell (editor)
AS207960 Cyfngedig
13 Pen-y-elan Terrace
Caerdydd
CF23 9EU
United Kingdom

Email: g@as207970.net, g@magicalcodewit.ch
URI: https://magicalcodewit.ch