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

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 27 juin 2025

Date de publication du RFC : Juin 2025

https://www.bortzmeyer.org/9799.html

Le protocole ACME, normalisé dans le RFC 8555[1], permet d'automatiser le processus de création et de renouvellement de certificats utilisables, par exemple, pour TLS. L'extension normalisée dans ce nouveau RFC permet d'obtenir et de renouveler des certificats pour un service utilisant le .onion de Tor. Si vous voulez passer à la télévision en disant « j'ai obtenu un certificat pour le Dark Web », ce RFC est la bonne lecture.

Ces services en .onion sont décrits dans la spécification de Tor <https://spec.torproject.org/>. Le nom .onion a été enregistré comme « nom spécial » (RFC 6761) par le RFC 7686. Ces noms en .onion ne sont pas résolvables via le DNS et on ne peut donc pas utiliser les défis ACME (RFC 8555, section 8) classiques avec une AC habituelle.

La norme ACME, le RFC 8555, définit dans sa section 9.7.7 des types d'**identificateurs**. Pour les .onion, ce sera le type dns, même si Tor n'utilise pas le DNS. J'aurais préféré que ce type se nomme domainname, ça aurait été plus cohérent (voir l'annexe A du RFC, qui discute ce choix, qui prend place dans un cadre très embrouillé, avec beaucoup de gens qui confondent noms de domaine et DNS). L'identificateur est le nom dans .onion, éventuellement avec des sous-domaines. Par exemple, le JSON envoyé par ACME, pour la version en .onion de ce blog <https://www.bortzmeyer.org/blog-tor-onion.html>, serait :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc8555.txt

```
{
  "type": "dns",
  "value": "sjnrk23rmcl4ie5atmz664v7o7k5nkk4jh7mm6lor2n4hxz2tos3eyid.onion"
}
```

Reste à permettre au serveur ACME (l'AC) de le valider. Le forum AC/Navigateur permet les `.onion` (annexe B.2 de ses « *"Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates"* <https://cabforum.org/working-groups/server/baseline-requirements/documents/CA-Browser-Forum-TLS-BR-2.0.6.pdf> »). ACME utilise le concept de **défi** pour ces validations. Le client ACME demande un certificat pour un certain identificateur, le serveur lui renvoie un défi, une tâche à accomplir, du genre « prouve-moi que tu peux mettre ces données que je t'envoie dans un endroit accessible en HTTP via le nom pour lequel tu veux un certificat ». Il existe plusieurs défis utilisables avec les `.onion` :

— `http-01` (section 8.3 du RFC 8555), notre RFC dit qu'il faut suivre les redirections HTTP, même si elles mènent à un serveur HTTP qui n'est pas en `.onion` ; c'est ce défi que j'ai utilisé pour mon blog.
— `tls-alpn-01` (RFC 8737).
— `onion-csr-01`, qui, contrairement aux deux précédents, est spécifique aux `.onion` et normalisé dans ce RFC 9799. C'est aussi le seul des trois défis utilisables qui permettre des jokers dans l'identificateur (`*.bbcweb3hytmzhn5d532owbu6oqadra5z3ar726vq5kgwwn6aucdccrad.onion`). Je le décris en détail plus loin. Cette méthode de validation figure désormais dans le registre des types de défis <https://www.iana.org/assignments/acme/acme.xml#acme-validation-methods
— `dns-01` (section 8.4 du RFC 8555), par contre, ne peut pas être utilisé pour les `.onion`, ceux-ci ne se servant pas du DNS.

Le défi `onion-csr-01` consiste à demander au client de génerer une CSR (RFC 2986) signée avec la clé privée du nom en `.onion` (je rappelle qu'un nom en `.onion` est une clé publique, plus exactement est dérivé d'une clé publique). Pour cela, le serveur ACME envoie un numnique <https://www.bortzmeyer.org/nonce.html> et sa clé publique Ed25519, qu'il utilise pour Tor. Le client doit répondre avec le CSR, signé avec sa clé privée, CSR contenant parmi ses attributs le numnique choisi par le serveur, et un autre choisi par le client. Le serveur vérifie alors ce CSR (le nom en `.onion`, et bien sûr la signature).

Notez qu'on peut imaginer une AC qui ne soit elle-même accessible que via un nom en `.onion`, ce qui serait cohérent (section 5 du RFC). Par contre, en sortie, une AC doit utiliser Tor pour se connecter aux `.onion` (je veux dire l'utiliser directement, pas en passant par une passerelle, cf. section 8.8) mais une AC ne doit **pas** utiliser Tor pour se connecter à des domaines non-`onion` (section 8.4) car il y a des nœuds de sortie méchants <https://link.springer.com/epdf/10.1007/978-3-319-08506-7_16?sharing_token=PRqtO-0qZN0jVqTGXJ1Y2_e4RwlQNchNByi7wbcMAY7YusGKWOOhuNdoJjCuIjBpCKwGEDJGV4qrueUPqTjbC4Ui3QdKJoMTJIKKM3shqmpGDwxbkTWjo2cJa3L7S3_mOkSAL2dbuhxhjeI%3D>.

Et les enregistrements de type CAA, normalisés dans le RFC 8659, et qui augmentent la sécurité du système en indiquant à quelles AC le client ACME a recours ? On ne peut pas utiliser le CAA traditionnel puisqu'il dépend du DNS. La solution, décrite dans la section 6 du RFC, est de mettre cette information dans le descripteur de service Tor <https://spec.torproject.org/dir-spec/server-descriptor-format.html> (celui que le serveur en `.onion` envoie aux serveurs d'annuaire Tor).

Passons à la pratique. Il existe une AC **expérimentale** (et qui ne prétend pas offrir les garanties de sécurité d'une « vraie »), par l'auteur du RFC, documentée sur le site Web du projet <https://acmeforonions.org>. Je m'en suis servi pour obtenir un certificat pour mon blog <https://www.bortzmeyer.org/blog-tor-onion.html>. Vous pouvez donc désormais vous connecter en

HTTPS, à (votre navigateur va sans doute râler car il ne connait pas cette AC). Attention, l'AC a enregistré ce certificat via "*Certificate Transparency*" (RFC 9162) et est donc publiquement visible <https://crt.sh/?id=16869821351>. Ne demandez pas de certificat à une AC qui utilise "*Certificate Transparency*" si ça vous dérange (section 8.9 du RFC).

Quelle est la marche à suivre pour obtenir un tel certificat ? D'abord, configurer Tor pour accepter le port de HTTPS :

```
  HiddenServiceDir /var/lib/tor/blogv2/
  HiddenServicePort 80 127.0.0.1:80
  HiddenServicePort 443 127.0.0.1:443
```

Ensuite, configurer son serveur HTTP, Apache dans mon cas :

```
# Pour le client ACME (qui écoutera sur le port 8080) :
ProxyPass /.well-known/acme-challenge http://localhost:8080/.well-known/acme-challenge
ProxyPassReverse /.well-known/acme-challenge http://localhost:8080/.well-known/acme-challenge
ProxyPreserveHost on
# Ne pas oublier d'activer mod_proxy *et* mod_proxy_http.

# HTTPS :
<VirtualHost _default_:443>
   ServerName sjnrk23rmcl4ie5atmz664v7o7k5nkk4jh7mm6lor2n4hxz2tos3eyid.onion

   # Liens symboliques vers les certificats obtenus par ACME, dans mon
   # cas /etc/letsencrypt/live/sjnrk23rmcl4ie5atmz664v7o7k5nkk4jh7mm6lor2n4hxz2tos3eyid.onion/...
   SSLCertificateFile /etc/apache2/server.pem
   SSLCertificateKeyFile /etc/apache2/server.key
```

On lance ensuite le client ACME. J'ai utilisé certbot <https://github.com/certbot/certbot> :

```
% sudo certbot certonly -v --server https://acme.api.acmeforonions.org/directory \
             --standalone --http-01-port 8080 --http-01-address 127.0.0.1 \
             -d sjnrk23rmcl4ie5atmz664v7o7k5nkk4jh7mm6lor2n4hxz2tos3eyid.onion
...
Performing the following challenges:
http-01 challenge for sjnrk23rmcl4ie5atmz664v7o7k5nkk4jh7mm6lor2n4hxz2tos3eyid.onion
Waiting for verification...
Cleaning up challenges
Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/sjnrk23rmcl4ie5atmz664v7o7k5nkk4jh7mm6lor2n4hxz2tos3eyid.onion/fu
Key is saved at:         /etc/letsencrypt/live/sjnrk23rmcl4ie5atmz664v7o7k5nkk4jh7mm6lor2n4hxz2tos3eyid.onion/pr
This certificate expires on 2025-05-28.
...
```

(Et pensez à configurer le renouvellement automatique du certificat, normalement, certbot l'a fait tout seul, mais vérifiez.)

Un tcpdump montre le trafic échangé entre Apache (qui a reçu le défi de l'AC, via Tor) et certbot (qui avait envoyé la demande et stocké le défi du serveur ACME), sur ma machine :

```
127.0.0.1.60978 > 127.0.0.1.8080: Flags [P.] ... HTTP, length: 376
 GET /.well-known/acme-challenge/G4JwaWsRDG42H_FAESGCQrY7ZYk3D3mY9Ob_j458z6M HTTP/1.1
 Host: sjnrk23rmcl4ie5atmz664v7o7k5nkk4jh7mm6lor2n4hxz2tos3eyid.onion
 X-Forwarded-For: 127.0.0.1
 X-Forwarded-Host: sjnrk23rmcl4ie5atmz664v7o7k5nkk4jh7mm6lor2n4hxz2tos3eyid.onion
 X-Forwarded-Server: sjnrk23rmcl4ie5atmz664v7o7k5nkk4jh7mm6lor2n4hxz2tos3eyid.onion

127.0.0.1.8080 > 127.0.0.1.60978: Flags [P.] ... HTTP, length: 92
 HTTP/1.0 200 OK
 Server: BaseHTTP/0.6 Python/3.11.2
 Date: Thu, 27 Feb 2025 17:04:53 GMT
```

Si vous voulez ajouter cette AC (mais rappelez-vous qu'elle est expérimentale et sans garanties de sécurité) au Tor Browser, vous devrez sans doute, dans `about:config`, configurer, **avant** d'importer l'AC :

```
security.nocertdb false
browser.privatebrowsing.autostart false
security.ssl.enable_ocsp_stapling false
```

Mais je répète mon avertissement : cela peut affecter votre vie privée. Ne le faites pas si vous ne comprenez pas en détail les conséquences.

J'ai utilisé le défi `http-01`, avec un certbot ordinaire. Le défi `onion-01` nécessite un certbot modifié (du code est disponible <`https://pypi.org/project/certbot-onion/`>).

Petite anecdote : l'auteur du RFC travaille pour l'AS 207960, dont l'objet RIPE annonce :

```
aut-num:        AS207960
as-name:        AS-TRANSRIGHTS
descr:          Trans Rights! Hell Yeah!
```

Et le site Web <`https://as207960.net/`> montre d'ailleurs le drapeau correspondant. Il est assez rare de voir des prises de position politiques dans la base des RIR.