RFC 8737 : Automated Certificate Management Environment (ACME) TLS Application-Layer Protocol Negotiation (ALPN) Challenge Extension

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 mars 2020

Date de publication du RFC : Février 2020

https://www.bortzmeyer.org/8737.html

Ce court RFC normalise un mécanisme d'authentification lors d'une session ACME, permettant de prouver, via TLS et ALPN, qu'on contrôle effectivement le domaine pour lequel on demande un certificat.

Le protocole ACME (RFC 8555 [1]) laisse le choix entre plusieurs mécanismes d'authentification pour répondre aux défis du serveur ACME « prouvez-moi que vous contrôlez réellement le domaine pour lequel vous me demandez un certificat ! ». Le RFC 8555 propose un défi fondé sur HTTP (`http-01`), dans sa section 8.3, et un défi utilisant le DNS (`dns-01`), dans sa section 8.4. Notez que le défi HTTP est fait en clair, sans HTTPS. Or, outre la sécurité de TLS, certains utilisateurs d'ACME auraient bien voulu une solution purement TLS, notamment pour les cas où la terminaison de TLS et celle de HTTP sont faites par deux machines différentes (CDN, répartiteurs de charge TLS, etc.)

D'où le nouveau défi normalisé par ce RFC, `tls-alpn-01`. Il repose sur le mécanisme ALPN, qui avait été normalisé dans le RFC 7301. Déjà mis en œuvre dans des AC comme Let's Encrypt, il permet une vérification plus solide. Ce type de défi figure maintenant dans le registre des types de défis ACME <https://www.iana.org/assignments/acme/acme.xml#acme-validation-methods>. Notez qu'il existait déjà un type utilisant TLS, `tls-sni-01 / tls-sni-02`, mais qui avait des failles <https://labs.detectify.com/2018/01/12/how-i-exploited-acme-tls-sni-01-issuing-lets-encry>, autorisant un utilisateur d'un serveur TLS à obtenir des certificats pour d'autres domaines du même serveur. `tls-sni` est aujourd'hui abandonné.

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

Les détails du mécanisme figurent dans la section 3 de notre RFC. Le principe est que le serveur ACME se connectera en TLS au nom de domaine indiqué en envoyant l'extension ALPN avec le nom d'application `acme-tls/1` et vérifiera dans le certificat la présence d'un "*token*", choisi aléatoirement par le serveur ACME, "*token*" que le client ACME avait reçu sur le canal ACME. (Ce nom d'application, `acme-tls/1` est désormais dans le registre des applications ALPN <https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xml#alpn-protocol-ids>.)

Bien sûr, c'est un peu plus compliqué que cela. Par exemple, comme le **client** ACME va devenir le **serveur** TLS lors du défi, il lui faut un certificat. La section 3 du RFC explique les règles auxquelles doit obéir ce certificat :

— Auto-signé, puisqu'on n'est pas encore authentifié auprès de l'AC,
— Un `subjectAlternativeName` (RFC 5280) qui a comme valeur le nom de domaine à valider,
— Une extension `acmeIdentifier` (mise dans le registre des extensions aux certificats PKIX <https://www.iana.org/assignments/smi-numbers/smi-numbers.xml#smi-numbers-1.3.6.1.5.5.7.1>), qui doit être marquée comme critique, pour éviter que des clients TLS passant par là et n'ayant rien à voir avec ACME s'en servent, et dont la valeur est l'autorisation ACME (RFC 8555, section 8.1).

Le client ACME doit ensuite configurer ce qu'il faut sur son serveur TLS pour que ce soit ce certificat qui soit retourné lors d'une connexion TLS où le SNI vaut le domaine à valider et où ALPN vaut `acme-tls/1`. Il annonce alors au serveur ACME qu'il est prêt à répondre au défi. Le serveur ACME se connecte au serveur TLS (créé par le client ACME) et fait les vérifications nécessaires (nom de domaine dans le certificat, nom qui doit être un "*A-label*", donc en Punycode, et extension du certificat `acmeIdentifier` contenant la valeur indiquée par le serveur ACME lors du défi).

Une fois la vérification faite, le serveur ACME raccroche : ces certificats ne sont pas conçus pour échanger de vraies données sur la session TLS créée. D'ailleurs, les certificats auto-signés créés pour le type de défi `tls-alpn-01` ne permettent pas d'authentification au sens du RFC 5280. Pour la même raison, le client TLS (créé par le serveur ACME) n'est pas obligé d'aller jusqu'au bout de l'établissement de la session TLS.

La section 5 de notre RFC fait le point sur quelques suppositions faites au sujet de la mise en œuvre de TLS, suppositions importantes pour ACME. D'abord, si plusieurs organisations ou personnes partagent la même adresse IP, ce qui est courant en cas d'hébergement plus ou moins mutualisé, ACME compte bien que leurs configurations TLS soient séparées, pour éviter qu'une de ces entités puisse obtenir un certificat pour une autre, hébergée au même endroit (cf. annexe A du RFC, qui décrit le comportement surprenant de certains hébergeurs.) ACME espère également que les serveurs TLS vont respecter le RFC 7301 en rejetant l'application `acme-tls/1` s'ils ne la connaissent pas. (Certains programmeurs paresseux ont peut-être fait en sorte que le serveur TLS accepte n'importe quelle application signalée en ALPN.)

L'AC Let's Encrypt accepte déjà ce type de défi <https://community.letsencrypt.org/t/tls-alpn-validation-method/63814> depuis juillet 2018. (Le RFC est en retard par rapport au déploiement effectif.) Le client ACME dehydrated <https://dehydrated.io/> permet d'utiliser le nouveau type de défi <https://github.com/dehydrated-io/dehydrated/blob/master/docs/tls-alpn.md>. Cet article <https://medium.com/@decrocksam/deploying-lets-encrypt-certi utilise nginx côté serveur, avec son module "*SSL PreRead*", qui permet d'aiguiller une requête en fonction de l'ALPN, mais, personnellement, je n'ai pas réussi (ça peut être un problème avec la gestion des modules dans le paquetage Debian de nginx, gestion quasiment pas documentée.)

Côté serveur, on a aussi ce qu'il faut dans Apache, avec le module `mod_md` (cf. plus précisement ce point de la documentation <https://github.com/icing/mod_md/#tls-alpn-challenges>.) Son utilisation est décrite dans un article de Marc Framboisier <https://www.framboisier.com/

blog/2021/08/14/apache-mod_md-et-lets-encrypt-simplissime/> (et sa suite <https://www.framboisier.com/blog/2021/08/19/acme-migration-certbot-vers-md-dapache/>).

Côté client ACME, d'autres clients <https://community.letsencrypt.org/t/which-client-support-tls-75859/2> gèrent ce type de défi, mais pas encore certbot <https://certbot.eff.org/> (cf. le ticket #6724 <https://github.com/certbot/certbot/issues/6724> .)