

RFC 6844 : DNS Certification Authority Authorization (CAA) Resource Record

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 31 août 2017. Dernière mise à jour le 26 octobre 2017

Date de publication du RFC : Janvier 2013

<https://www.bortzmeyer.org/6844.html>

Ce RFC a quelques années mais n'avait connu que peu de déploiement effectif. Cela va peut-être changer à partir du 8 septembre 2017, où les AC ont promis de commencer à vérifier les enregistrements CAA ("*Certification Authority Authorization*") publiés dans le DNS. Ça sert à quoi ? En gros, un enregistrement CAA publié indique quelles AC sont autorisées à émettre des certificats pour ce domaine. Le but est de corriger très partiellement une des plus grosses faiblesses de X.509, le fait que n'importe quelle AC peut émettre des certificats pour n'importe quel domaine, même si ce n'est pas un de ses clients. Notez qu'il a depuis été remplacé par le RFC 8659¹.

C'est donc une technique très différente de DANE (RFC 6698), les seuls points communs étant l'utilisation du DNS pour sécuriser les certificats. DANE est déployé chez le client TLS, pour qu'il vérifie le certificat utilisé, CAA est surtout dans l'AC, pour limiter le risque d'émission d'un certificat malveillant (par exemple, CAA aurait peut-être empêché le faux certificat Gmail du ministère des finances <<http://www.01net.com/actualites/comment-le-ministere-des-finances-espionne-le-trafic-web-de-ses.html>>.) Disons que CAA est un contrôle supplémentaire, parmi ceux que l'AC doit (devrait) faire. Les clients TLS ne sont pas censés le tester (ne serait-ce que parce que l'enregistrement CAA a pu changer depuis l'émission du certificat, la durée de vie de ceux-ci étant en général de plusieurs mois). CAA peut aussi servir à des auditeurs qui veulent vérifier les pratiques d'une AC (même avertissement : le certificat a pu être émis alors que l'enregistrement CAA était différent.)

La section 3 de notre RFC présente l'enregistrement CAA (la syntaxe détaillée est en section 5). Il a été ajouté au registre des types d'enregistrements <<https://www.iana.org/assignments/dns-parameters/dns-parameters.xml#dns-parameters-4>> sous le numéro 257. Il comprend

1. Pour voir le RFC de numéro NNN, <https://www.ietf.org/rfc/rfcNNN.txt>, par exemple <https://www.ietf.org/rfc/rfc8659.txt>

une série d'options ("*flags*") et une **propriété** qui est sous la forme {clé, valeur}. Un nom peut avoir plusieurs propriétés. Pour l'instant, une seule option est définie (un registre <<https://www.iana.org/assignments/pkix-parameters/pkix-parameters.xml#caa-flags>> existe pour les options futures), « "*issuer critical*" » qui indique que cette propriété est cruciale : si on ne la comprend pas, le test doit être considéré comme ayant échoué et l'AC ne doit pas produire de certificat.

Les propriétés possibles sont, à l'heure actuelle (d'autres pourront être ajoutée dans le registre IANA <<https://www.iana.org/assignments/pkix-parameters/pkix-parameters.xml#caa-properties>>)

- *issue*, la principale, qui indique une AC autorisée à émettre des certificats pour ce domaine (l'AC est indiquée par son nom de domaine),
- *issuewild*, idem, mais avec en plus la possibilité pour l'AC d'émettre des certificats incluant des jokers,
- *iodef*, qui indique où l'AC doit envoyer d'éventuels rapports d'échec, pour que le titulaire du nom de domaine puisse les corriger. Un URL est indiqué pour cela, et le rapport doit être au format IODEF (RFC 7970).

Voici par exemple quel était l'enregistrement CAA de mon domaine personnel :

```
% dig CAA bortzmeyer.org
...
;; -->HEADER<<- opcode: QUERY, status: NOERROR, id: 61450
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 3, AUTHORITY: 7, ADDITIONAL: 7
...
;; ANSWER SECTION:
bortzmeyer.org. 26786 IN CAA 0 issue "cacert.org"
bortzmeyer.org. 26786 IN CAA 0 issuewild "\;"
...
```

Il indique que seule l'AC CACert <<https://www.bortzmeyer.org/cacert.html>> peut créer un certificat pour ce domaine (et sans les jokers). Bon, c'est un peu inutile car CACert ne teste pas les enregistrements CAA, mais c'était juste pour jouer. Je n'ai pas mis d'*iodef* mais il aurait pu être :

```
bortzmeyer.org. CAA 0 iodef "mailto:security@bortzmeyer.org"
```

Et, dans ce cas, l'AC peut écrire à security@bortzmeyer.org, avec le rapport IODEF en pièce jointe.

Attention, l'enregistrement CAA est valable pour tous les sous-domaines (et ce n'est pas une option, contrairement à, par exemple, HSTS du RFC 6797, avec son *includeSubDomains*). C'est pour cela que j'ai dû retirer l'enregistrement ci-dessus, pour avoir des certificats pour les sous-domaines, certificats faits par une autre AC.

Des paramètres peuvent être ajoutés, le RFC cite l'exemple d'un numéro de client :

```
example.com. CAA 0 issue "ca.example.net; account=230123"
```

<https://www.bortzmeyer.org/6844.html>

Une fois les enregistrements CAA publiés, comment sont-ils utilisés (section 4)? L'AC est censée interroger le DNS pour voir s'il y a un CAA (on note que DNSSEC est recommandé, mais n'est pas obligatoire, ce qui réduit le service déjà faible qu'offre CAA). S'il n'y en a pas, l'AC continue avec ses procédures habituelles. S'il y a un CAA, deux cas : il indique que cette AC peut émettre un certificat pour le domaine, et dans ce cas-là, c'est bon, on continue avec les procédures habituelles. Second cas, le CAA ne nomme pas cette AC et elle doit donc renoncer à faire un certificat **sauf** s'il y a une exception configurée pour ce domaine (c'est la deuxième faille de CAA : une AC peut facilement passer outre et donc continuer émettre de « vrais/faux certificats »).

Notez que le RFC ne semble pas évoquer la possibilité d'imposer la présence d'un enregistrement CAA. C'est logique, vu le peu de déploiement de cette technique mais cela veut dire que « qui ne dit mot consent ». Pour la plupart des domaines, la vérification du CAA par l'AC ne changera rien.

Notez que, si aucun enregistrement CAA n'est trouvé, l'AC est censé remonter l'arbre du DNS. (C'est pour cela que SSL [sic] Labs <<https://www.ssllabs.com/>> trouve un enregistrement CAA pour `mercredifiction.bortzmeyer.org` : il a utilisé le CAA du domaine parent, `bortzmeyer.org`.) Si `example.com` n'a pas de CAA, l'AC va tester `.com`, demandant ainsi à Verisign si son client peut avoir un certificat et de qui. Cette erreur consistant à grimper sur l'arbre avait déjà été dénoncée dans le RFC 1535, mais apparemment la leçon n'a pas été retenue.

Enfin, la section 6 du RFC analyse les différents problèmes de sécurité que peut poser CAA :

- Le respect de l'enregistrement CAA dépend de la bonne volonté de l'AC (et CAA ne remplace donc pas DANE),
- Sans DNSSEC, le test CAA est vulnérable à des attaques par exemple par empoisonnement (le RFC conseille de ne pas passer par un résolveur normal mais d'aller demander directement aux serveurs faisant autorité, ce qui ne résout pas le problème : le programme de test peut se faire empoisonner, lui aussi, d'autant plus qu'il prend sans doute moins de précautions qu'un résolveur sérieux),

— Et quelques autres risques plus exotiques.

Comme indiqué au début, le CA/Browser Forum a décidé que le test des CAA serait obligatoire à partir du 8 septembre 2017. (Cf. la décision <<https://cabforum.org/2017/03/08/ballot-187-make-caa-check>>.) Parmi les rares enregistrements CAA dans la nature, on trouve celui de Google, qui autorise deux AC :

```
% dig CAA google.com
...
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55244
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
...
;; ANSWER SECTION:
google.com. 86400 IN CAA 0 issue "pki.goog"
google.com. 86400 IN CAA 0 issue "symantec.com"
...
```

(Le TLD `.goog` est apparemment utilisé par Google pour son infrastructure, `.google` étant plutôt pour les choses publiques.) Notez que `gmail.com`, lui, pourtant souvent détourné par des gouvernements et des entreprises qui veulent surveiller le trafic, n'a pas d'enregistrement CAA. Le célèbre SSL [sic] Labs <<https://www.ssllabs.com/>> teste la présence d'un enregistrement CAA. S'il affiche « "DNS Certification Authority Authorization (CAA) Policy found for this domain" », c'est bon. Regardez le cas de Google <<https://www.ssllabs.com/ssltest/analyze.html?d=google.com&s=172.217.5.110>>.

Le type CAA est relativement récent (quatre ans quand même), donc il crée des problèmes amusants avec les serveurs DNS bogués comme ceux de BNP Paribas <<https://www.bortzmeyer.org/panne-dns-bnpparibas.html>>, qui "timeoutent" lorsque le type demandé n'est pas A :

```
% dig CAA mabanqueprivée.bnpparibas
; <<>> DiG 9.10.3-P4-Debian <<>> CAA mabanqueprivée.bnpparibas
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 42545
                ^^^^^^^^
```

Quelques lectures et ressources pour finir :

- Un bon article de synthèse par SSL Labs <<https://blog.qualys.com/ssllabs/2017/03/13/caa-mandated-by-cabrowser-forum>>.
- Une aide interactive pour fabriquer un enregistrement CAA <<https://sslmate.com/labs/caa/>>. Si vous avez un très vieux serveur DNS, et qu'il refuse de charger les enregistrements CAA, vous allez devoir utiliser la syntaxe générique du RFC 3597. Comme elle est un peu compliquée, le service ci-dessus, ainsi que ce service en ligne <<https://anders.com/projects/sysadmin/djbdnsRecordBuilder/#CAA>> peuvent vous aider, en affichant à l'ancienne syntaxe.
- Cet article détaillé écrit par une AC <<https://www.digicert.com/blog/new-caa-requirement-2/>> note un grand nombre de problèmes et de limites de CAA.
- Dès le lendemain du début des tests CAA par les AC, Comodo (la société des auteurs du RFC) avait déjà violé les règles <<https://www.bleepingcomputer.com/news/security/comodo-caught-b>> et émis un certificat qui n'aurait pas dû l'être <<https://crt.sh/?id=207082245>>.
- Let's Encrypt vérifie bien les enregistrements CAA. Si on a un CAA indiquant une autre AC, même pour un domaine parent du domaine pour lequel on veut un certificat, Let's Encrypt refuse avec une erreur de type `urn:acme:error:caa`, « *CAA record for DOMAIN_NAME prevents issuance* ».