RFC 9162 : Certificate Transparency Version 2.0

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 10 décembre 2021

Date de publication du RFC : Décembre 2021

https://www.bortzmeyer.org/9162.html

Le système de gestion de certificats PKIX (dérivé des certificats X.509) a une énorme faiblesse. N'importe quelle AC peut émettre un certificat pour n'importe quel nom de domaine. Il ne suffit donc pas de bien choisir son AC, votre sécurité dépend de **toutes** les AC. Ce RFC propose une approche pour combler cette faille de sécurité : encourager/obliger les AC à **publier** « au grand jour » les certificats qu'elles émettent. Un titulaire d'un certificat qui craint qu'une AC n'émette un certificat à son nom sans son autorisation n'a alors qu'à surveiller ces publications. (Il peut aussi découvrir à cette occasion que sa propre AC s'est fait pirater ou bien est devenue méchante et émet des certificats qu'on ne lui a pas demandés. L'idée est aussi d'empêcher l'émission « discrète » de vrais/faux certificats qui seraient ensuite utilisés uniquement à certains endroits.) Ce système, dit « "*Certificate Transparency*" <https://certificate.transparency.dev/> » (CT) avait initialement été normalisé dans le RFC 6962 [1], que notre RFC remplace, le protocole passant à une nouvelle version, la v2 (toujours considérée comme expérimentale).

Le principe est donc de créer des journaux des certificats émis. Le journal doit être public, pour que n'importe qui puisse l'auditer (section 4 du RFC). Il doit être en mode « ajout seulement » pour éviter qu'on puisse réécrire l'histoire. Les certificats sont déjà signés mais le journal a ses propres signatures, pour prouver son intégrité. Conceptuellement, ce journal est une liste de certificats dans l'ordre de leur création. Toute AC peut y ajouter des certificats (la liste ne peut pas être ouverte en écriture à tous, de crainte qu'elle ne soit remplie rapidement de certificats bidons). En pratique, le RFC estime que la liste des AC autorisées à écrire dans le journal sera l'union des listes des AC acceptées dans les principaux navigateurs Web (voir aussi les sections 4.2 et 5.7, chaque journal est responsable de ce qu'il accepte ou pas comme soumissions).

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6962.txt

À chaque insertion, le journal renvoie à l'AC une estampille temporelle signéee (SCT, pour "*Signed Certificate Timestamp*"), permettant à l'AC de prouver qu'elle a bien enregistré le certificat. Si on a cette signature mais que le certificat est absent du journal, l'observateur aura la preuve que le journal ne marche pas correctement. Le format exact de cette estampille temporelle est décrit en section 4.8. Idéalement, elle devra être envoyée au client par les serveurs TLS, dans l'extension TLS `transparency_info` (désormais enregistrée à l'IANA <`https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xml#tls-extensiontype-values-1`>), comme preuve de la bonne foi de l'AC (cf. section 6 et notamment 6.5, car c'est plus compliqué que cela). Bien sûr, cette validation de l'insertion dans un journal ne dispense pas de la validation normale du certificat (un certificat peut être journalisé et mensonger à la fois). Notez aussi que, si le serveur TLS n'envoie pas toutes les données au client, celui-ci peut les demander au journal (opérations `/get-proof-by-hash` et `get-all-by-hash`) mais, ce faisant, il informe le journal des certificats qui l'intéressent et donc, par exemple, des sites Web qu'il visite.

De même, une extension à OCSP (RFC 6960) peut être utilisée pour appuyer les réponses OCSP. On peut même inclure les preuves d'inclusion dans le journal dans le certificat lui-même, ce qui permet d'utiliser des serveurs TLS non modifiés.

Les titulaires de certificats importants, comme Google, mais aussi des chercheurs, des agences de sécurité, etc, pourront alors suivre l'activité de ces journaux publics (section 8.2 du RFC). Ce qu'ils feront en cas de détection d'un certificat anormal (portant sur leur nom de domaine, mais qu'ils n'ont pas demandé) n'est pas spécifié dans le RFC : cela dépend de la politique de l'organisation concernée. Ce RFC fournit un mécanisme, son usage n'est pas de son ressort. Ce journal n'empêchera donc pas l'émission de vrais/faux certificats, ni leur usage, mais il les rendra visibles plus facilement et sans doute plus vite.

Notons que les certificats client, eux, ne sont typiquement pas journalisés (rappelez-vous que les journaux sont publics et que les certificats client peuvent contenir des données personnelles). Le serveur TLS ne peut donc pas utiliser "*Certificate Transparency*" pour vérifier le certificat du client. (Le RFC estime que le principal risque, de toute façon, est celui d'usurpation du serveur, pas du client.)

Pour que cela fonctionne, il faudra que les clients TLS vérifient que le certificat présenté est bien dans le journal (autrement, le méchant n'aurait qu'à ne pas enregistrer son vrai/faux certificat, cf. section 8.3 du RFC).

En pratique, la réalisation de ce journal utilise un arbre de Merkle, une structure de données qui permet de mettre en œuvre un système où l'ajout de certificats est possible, mais pas leur retrait, puisque chaque nœud est un condensat de ses enfants (voir aussi le RFC 8391). La section 2 du RFC détaille l'utilisation de ces arbres et la cryptographie utilisée. (Et les exemples en section 2.1.5 aident bien à comprendre comment ces arbres de Merkle sont utilisés.)

Le protocole utilisé entre les AC et le journal, comme celui utilisé entre les clients TLS et le journal, est HTTP et le format des données du JSON (section 5, qui décrit l'API). Ainsi, pour ajouter un certificat nouvellement émis au journal géré sur `sunlight-log.example.net`, l'AC fera :

```
POST https://sunlight-log.example.net/ct/v2/submit-entry
```

et le corps de la requête HTTP sera un tableau JSON de certificats encodés en Base64. La réponse contiendra notamment l'estampille temporelle (SCT pour "*Signed Certificate Timestamp*"). S'il y a un problème, le client recevra une des erreurs enregistrées <`https://www.iana.org/assignments/trans/trans.xml#trans-error-types`>. Pour récupérer des certificats, le programme de surveillance fera par exemple :

```
GET https://sunlight-log.example.net/ct/v2/get-entries
```

D'autres URL permettront de récupérer les condensats cryptographiques contenus dans l'arbre de Merkle, pour s'assurer qu'il est cohérent.

Comme il n'existe (en octobre 2021) aucune mise en œuvre de la version 2 du protocole, voici quelques exemples, utilisant des journaux réels, et la version 1 du protocole (notez le `v1` dans l'URL). Pour trouver les coordonnées des journaux, j'ai utilisé la liste « officielle » du projet <http://www.certificate-transparency.org/known-logs>. Notez que tous les journaux qui y figurent ne fonctionnent pas correctement. Notez aussi que, comme pour les AC ou les serveurs de clés PGP, il n'y a pas de « journal de référence », c'est à chacun de choisir les journaux où il va écrire, et qu'il va lire. Le script (en ligne sur `https://www.bortzmeyer.org/files/test-ct-logs-v1.py`) teste la liste, et trouve :

```
50 logs are OK, 54 are currently broken
```

Si vous vous demandez pourquoi un même opérateur a plusieurs journaux, c'est en partie parce qu'il n'est pas possible de faire évoluer les algorithmes cryptographiques au sein d'un même journal (cf. section 9 du RFC) et qu'il faut donc de temps en temps créer un nouveau journal. Un journal est identifié par son URL, sa clé publique et (en v2) par son OID. Par exemple, le journal « Nimbus 2021 » de Cloudflare est en `https://ct.cloudflare.com/logs/nimbus2021/` et a la clé `MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAExpon7ipsqehIeU1bmpog9TFo4Pk8+9oN8OYHl1Q2JGVXnkVFnuuvPg` (je ne donne pas l'OID, il n'existe pas encore de journaux qui utilisent la version 2 du protocole). Voici un exemple d'utilisation (le STH est le "*Signed Tree Head*", la valeur de la racine de l'arbre de Merkle, cf. section 4.2.10) :

```
% curl -s https://ct.cloudflare.com/logs/nimbus2021/ct/v1/get-sth | jq .
{
  "tree_size": 408013312,
  "timestamp": 1634739692384,
  "sha256_root_hash": "7GnGjI7L6O5fn8kQKTdJG2riShNTTbcjRP2WbLoZrvQ=",
  "tree_head_signature": "BAMARjBEAiAQ0gb6udc9e28ykUGUzl0HV8U5NlJhPVSTUF4JtXGSeQIgcSbZ9kRgttGzpFETFem4eCv7GgUYPU
}
```

Plus de quatre cents millions de certificats, fichtre. Si on veut récupérer les deux premiers certificats journalisés :

```
% curl -s https://ct.cloudflare.com/logs/nimbus2021/ct/v1/get-entries\?start=0\&end=1 | jq .
{
  "entries": [
    {
      "leaf_input":
    [L'exemple est fait avec un journal v1, l'objet JSON renvoyé est
    différent en v2.]
```

Mais vous pouvez aussi utiliser "*Certificate Transparency*" (CT) sans aller regarder du JSON. Un service en ligne comme vous permet de scruter un journal. Voici par exemple l'actuel certificat de ce blog <https://crt.sh/?id=5169878041>, ou bien tous les certificats au nom de la Banque Postale <https://crt.sh/?q=labanquepostale.fr> (CT est utile pour le renseignement).

On a vu que plusieurs acteurs intervenaient, le gérant du journal, les AC, les gens qui regardent le journal, par exemple pour l'auditer, etc. Une utilisation courante de CT est pour surveiller l'émission de certificats au nom de son entreprise ou de son organisation, pour repérer les AC qui créent des certificats incorrects. Pour éviter de programmer tout cela de zéro en partant du RFC, on peut utiliser le service Certstream, qui sert d'intermédiaire avec plusieurs journaux, et sa bibliothèque Python <https://github.com/CaliDog/certstream-python/>. Ainsi, le petit script (en ligne sur https://www.bortzmeyer.org/files/test-certstream.py) permet de détecter tous les certificats émis pour les noms de domaine en .fr :

```
% pip3 install certstream

% ./test-certstream.py
...
[2021-10-23T13:21:46] pimpmydrone.fr (SAN: www.pimpmydrone.fr)
[2021-10-23T13:21:51] pascal-goldbach.fr (SAN: www.pascal-goldbach.fr)
[2021-10-23T13:21:52] leginkobiloba.fr (SAN: www.leginkobiloba.fr)
[2021-10-23T13:21:52] promabat-distribution.fr (SAN: www.promabat-distribution.fr)
[2021-10-23T13:21:53] maevakaliciak.fr (SAN: mail.maevakaliciak.fr, www.maevakaliciak.fr)
[2021-10-23T13:21:55] pascal-goldbach.fr (SAN: www.pascal-goldbach.fr)
[2021-10-23T13:21:56] maevakaliciak.fr (SAN: mail.maevakaliciak.fr, www.maevakaliciak.fr)
[2021-10-23T13:21:57] blog.nicolas-buffart.itval.fr (SAN: euromillions-generator.itval.fr, itval.fr, loto-ge
...
```

Bien sûr, cela fait beaucoup (regardez les intervalles entre deux messages). En pratique, on modifierait sans doute ce script pour ne regarder que les noms de son organisation. Ainsi, vous pouvez détecter les certificats et chercher ensuite s'ils sont légitimes (ce qui, dans certaines organisations très cloisonnées n'ira pas de soi…).

À part Certstream, Samuel Bizien Filippi me suggère CertSpotter <https://sslmate.com/certspotter/> mais qui me semble uniquement payant. Il a une API <https://certspotter.com/>. Elle peut être utilisée par le programme check_ct_logs <https://github.com/Samuel-BF/check_ct_logs>, qui peut être utilisé comme script de test pour les programmes de supervision comme Icinga.

Le projet « "*Certificate Transparency*" » (largement impulsé par Google) a un site officiel <https://certificate.transparency.dev/> (lecture recommandée) et, une liste de diffusion <https://groups.google.com/g/certificate-transparency> (sans compter le groupe de travail IETF TRANS <https://datatracker.ietf.org/wg/trans/>, mais qui se limitait à la normalisation, il ne parle pas des aspects opérationnels, et il a de toute façon été clos en août 2021). Questions logiciels, si vous voulez créer votre propre journal, il y a le programme de Google <https://github.com/google/certificate-transparency>.

Aujourd'hui, on peut dire que « "*Certificate Transparency*" » est un grand succès. La plupart (voire toutes ?) des AC y participent, il existe de nombreux journaux publics <http://www.certificate-transparency.org/known-logs>, et ils sont fréquemment utilisés pour l'investigation numérique (voire pour le renseignement, puisqu'on peut savoir, via les journaux, les noms de domaine pas encore annoncés, ce qui a parfois été cité comme une objection contre CT). Un bon exemple est celui de l'attaque « moyen-orientale » de 2018 <https://www.bortzmeyer.org/dnspionage.html> (mais il y a aussi l'affaire du certificat révoqué de la Poste <https://www.bortzmeyer.org/poste-revocation.html>).

Par contre, un client TLS ne peut pas encore être certain que tous les certificats seront dans ces journaux, et ne peut donc pas encore refuser les serveurs qui ne signalent pas la journalisation du certificat. Et le navigateur Firefox ne teste pas encore la présence des certificats dans le sjournaux <https://bugzilla.mozilla.org/show_bug.cgi?id=1281469>.

Un point amusant : le concept de « "*Certificate Transparency*" » montre qu'il est parfaitement possible d'avoir un livre des opérations publiquement vérifiable sans chaine de blocs. La chaine de blocs reste nécessaire quand on veut autoriser l'écriture (et pas juste la lecture) au public.

La section 1.3 du RFC résume les principaux changements entre les versions 1 (RFC 6962) et 2 du protocole :

— Les algorithmes cryptographiques utilisés sont désormais dans des registres IANA <https://www.iana.org/assignments/trans/trans.xml>,
— certains échanges utilisent désormais le format CMS (RFC 5652),
— les journaux qui étaient auparavant idenfiés par un condensat de leur clé publique le sont désormais par un OID, enregistrés à l'IANA <https://www.iana.org/assignments/trans/trans.xml#trans-log-ids> (aucun n'est encore attribué),
— nouvelle fonction `get-all-by-hash` dans l'API,
— remplacement de l'ancienne extension TLS `signed_certificate_timestamp` (valeur 18) par `transparency_info` (valeur 52, et voir aussi le nouvel en-tête HTTP `Expect-CT:` du RFC 9163,
— et d'autres changements, dans les structures de données utilisées.

CT ne change pas de statut avec la version 2 : il est toujours classé par l'IETF comme « Expérimental » (bien que largement déployé). La sortie de cette v2 n'est pas allée sans mal (le premier document étant sorti en février 2014), avec par exemple aucune activité du groupe pendant la deuxième moitié de 2020.

Une des plus chaudes discussions pour cette v2 avait été la proposition de changer l'API pour que les requêtes, au lieu d'aller à <BASE-URL>/ct/v2/ partent du chemin `/.well-known/` du RFC 8615. Cette idée a finalement été rejetée, malgré le RFC 8820, qui s'oppose à cette idée de chemins d'URL en dur.