RFC 9018 : Interoperable Domain Name System (DNS) Server Cookies

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 6 avril 2021

Date de publication du RFC : Avril 2021

https://www.bortzmeyer.org/9018.html

Le RFC 7873[1] normalisait un mécanisme, les "*cookies*", pour qu'un serveur DNS authentifie raisonnablement l'adresse IP du client (ce qui, normalement, n'est pas le cas en UDP <https://www.bortzmeyer.org/usurpation-adresse-ip.html>). Comme seul le serveur avait besoin de reconnaitre ses "*cookies*", le RFC 7873 n'imposait pas un algorithme particulier pour les générer. Cela posait problème dans certains cas et notre nouveau RFC propose donc un mécanisme standard pour fabriquer ces "*cookies*".

L'un des scénarios d'utilisation de ces « "*cookies*" standards » est le cas d'un serveur "*anycast*" dont on voudrait que toutes les instances génèrent des "*cookies*" standardisés, pour qu'une instance puisse reconnaitre les "*cookies*" d'une autre. Le RFC 7873, section 6, disait bien que toutes les instances avaient intérêt à partir du même secret mais ce n'était pas plus détaillé que cela. La situation actuelle, où chaque serveur peut faire différemment, est décrite dans le RFC par un mot en anglais que je ne connaissais pas, "*gallimaufry*" <https://en.wiktionary.org/wiki/gallimaufry>. D'où l'idée de spécifier cet algorithme, pour simplifier la vie des programmeuses et programmeurs avec un algorithme de génération de "*cookie*" bien étudié et documenté ; plus besoin d'en inventer un. Et cela permet de créer un service "*anycast*" avec des logiciels d'auteurs différents. S'ils utilisent cet algorithme, ils seront compatibles. Il ne s'agit que d'une possibilité : les serveurs peuvent utiliser cet algorithme mais ne sont pas obligés, le RFC 7873 reste d'actualité.

Comment, donc, construire un "*cookie*" ? Commençons par celui du client (section 3). Rappel : le but est d'authentifier un minimum donc, par exemple, le client doit choisir un "*cookie*" différent par serveur (sinon, un serveur pourrait se faire passer pour un autre, d'autant plus que le "*cookie*" circule en clair), et, donc, il faut utiliser quelque chose de spécifique au serveur dans l'algorithme, par exemple

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7873.txt

son adresse IP. Par contre, pas besoin de le changer souvent, il peut parfaitement durer des mois, sauf évidemment si un élément entrant dans sa construction change, ou est compromis. Autrefois, il était suggéré d'utiliser l'adresse IP du client dans la construction du "*cookie*" client, mais cette suggestion a été retirée car le logiciel ne connait parfois son adresse IP que trop tard (ça dépend de l'API réseau utilisée et, par exemple, avec l'API sockets de si on a déjà fait un `bind()` et, de toute façon, si on est derrière un routeur NAT, connaitre l'adresse IP locale ne sert pas à grand'chose). Néanmoins, pour éviter qu'un "*cookie*" ne permette à un observateur de relier deux requêtes d'une même machine, le "*cookie*" doit changer quand l'adresse IP change, comme rappelé par la section 8.1. (C'est particulièrement important si on utilise des techniques de protection de la vie privée comme celle du RFC 8981.)

Voilà, c'est tout parce que ce qui est important dans notre RFC, c'est le "*cookie*" serveur (section 4), puisque c'est là qu'on voudrait un "*cookie*" identique pour toutes les instances du service. Les éléments utilisés pour le générer sont le "*cookie*" du client, l'adresse IP du serveur, quelques métadonnées et un secret (qu'il faudra donc partager au sein du service "*anycast*"). Le secret changera, par exemple une fois par mois. Une fois qu'on a tous ces éléments, on va ensuite condenser le tout, ici avec SipHash (cf. J. Aumasson, et D. J. Bernstein, « "*SipHash : A Fast Short- Input PRF*" <https://cr.yp.to/siphash/siphash-20120918.pdf> »). (Parmi les critères de choix d'une fonction de condensation, il y a les performances, un serveur DNS actif pouvant avoir à faire ce calcul souvent, pour vérifier les "*cookies*".) Le "*cookie*" comprendra un numéro de version, un champ réservé, une estampille temporelle et le condensat. L'algorithme de génération du condensat inclus dans le "*cookie*" est donc : Condensat = SipHash-2-4 ("*Cookie*" client — Version — Réservé — Estampille — Client-IP, Secret), avec :

— Le signe — indique la concaténation.
— Le champ Version vaut actuellement 1 (les futures versions seront dans un registre IANA <https://www.iana.org/assignments/dns-parameters/dns-parameters.xml#server-cookie-metho
— Le champ Réservé ne comporte pour l'instant que des zéros.
— L'estampille temporelle sert à éviter les attaques par rejeu et permet de donner une durée de validité aux "*cookies*" (le RFC recommande une heure, avec cinq minutes de battement pour tenir compte des horloges mal synchronisées).

On a vu qu'il fallait changer le secret connu du serveur de temps en temps (voire plus rapidement s'il est compromis). Pour que ça ne casse pas tout (un serveur ne reconnaissant pas les "*cookies*" qu'il avait lui-même émis avec l'ancien secret…), il faut une période de recouvrement où le serveur connait déjà le nouveau secret (et accepte les "*cookies*" ainsi générés, par exemple par les autres instances du service "*anycast*"), puis une période où le serveur génère les "*cookies*" avec le nouveau secret, mais continue à accepter les "*cookies*" anciens (par exemple gardés par des clients). La section 5 rappelle, comme on le fait pour les remplacements de clés DNSSEC, de tenir compte des TTL pour calculer les délais nécessaires.

Si vous voulez mettre en œuvre vous-même cet algorithme, notez que l'annexe A du RFC contient des vecteurs de test, permettant de vérifier si vous ne vous êtes pas trompé.

La section 2 de notre RFC décrit les changements depuis le RFC 7873. Les algorithmes données comme simples suggestions dans les annexes A.1 et B.1 sont trop faibles et ne doivent pas être utilisés. Celui de l'annexe B.2 ne pose pas de problèmes de sécurité mais celui de notre nouveau RFC est préféré.

Plusieurs mises en œuvre de ce nouvel algorithme ont été faites, et leur interopérabilité testée (notamment au cours du hackathon de la réunion IETF 104 <https://www.ietf.org/how/meetings/104/> à Prague). Au moins BIND, Knot et getdns <https://getdnsapi.net/> ont déjà cet algorithme dans leurs versions publiées.