RFC 7128 : RPKI Router Implementation Report

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 22 février 2014

Date de publication du RFC : Février 2014

https://www.bortzmeyer.org/7128.html

Le protocole RTR ("*RPKI to Router Protocol*"), normalisé dans le RFC 6810 [1], est utilisé entre un routeur et un cache/validateur qui vérifie les objets signés de la RPKI. Ce nouveau RFC fait le point sur les mises en œuvre effectives de RTR et leur degré de couverture de la norme.

C'est un RFC un peu particulier : ce n'est pas la normalisation d'un protocole (c'est fait dans le RFC 6810) et ce n'est pas une étude indépendante, juste une collecte de réponses à un questionnaire. Les auteurs du RFC n'ont pas vérifié ces réponses, envoyées par les auteurs des différentes mises en œuvre de RTR.

Petit rappel sur la RPKI <https://www.bortzmeyer.org/securite-routage-bgp-rpki-roa.html> : les titulaires de ressources Internet (les préfixes d'adresses IP, notamment) émettent des objets cryptographiquement signés attestant notamment de la légitimité d'un AS à annoncer tel préfixe en BGP. La vérification de ces objets nécessite des calculs cryptographiques pour lesquels le routeur moyen n'est pas forcément bien équipé. L'architecture conseillée est donc d'avoir une machine Unix ordinaire, le cache/validateur, qui va faire ces calculs, et le routeur se contentera de charger les résultats depuis un cache/validateur de confiance. Entre les deux, le protocole RTR, où le serveur est le cache/validateur et le client est le routeur.

La section 2 du RFC liste les mises en œuvre interrogées, aussi bien clients que serveur. (J'ai écrit une bibliothèque très minimum pour faire des clients RTR en Go à des fins notamment de statistiques : GoRTR <https://github.com/bortzmeyer/GoRTR>. Très réduite et peu utilisée, elle ne fait pas partie de la liste étudiée par notre RFC.) On trouve donc dans ce RFC :

— Des clients RTR, aussi bien des routeurs (IOS, XR ou JunOS) que des bibliothèques permettant de bâtir des clients (RTRlib <http://rpki.realmv6.org/>).

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6810.txt

— Des serveurs RTR, deux en logiciel libre, rpki.net <http://rpki.net/> (rcynic) et le logiciel du RIPE-NCC <http://www.ripe.net/lir-services/resource-management/certification/tools-and-resources>, puis celui de BBN <http://sourceforge.net/projects/rpstir/> (rpstir).

Le reste du RFC contient les résultats du questionnaire. D'abord, la gestion des différents types de message (section 3). Tout le monde le fait, évidemment, sauf XR et JunOS qui ne gèrent pas encore les messages "*Error Report*". Ensuite, la gestion des échanges entre client et serveur (section 4) : presque complet sauf le programme du RIPE-NCC qui, aux questions "*Serial Query*", envoie toujours un "*Cache Reset*", forçant le client à tout télécharger.

Une des questions chaudes lors de la mise au point de RTR était le mécanisme de transport entre le client et le serveur, notamment pour assurer la sécurité (si le client, le routeur, parce qu'il a été trompé, parle à un serveur pirate, la sécurité fournie par la RPKI disparait, puisque le routeur ne fait pas les vérifications cryptographiques lui-même). Finalement, pour tenir compte des capacités très différentes des routeurs du marché, le RFC 6810 avait laissé les programmeurs choisir leur transport sécurisé librement. L'étude montre (section 5) que le seul protocole commun à toutes les mises en œuvre de RTR est le TCP pur, non sécurisé... Un autre choix, TCP protégé par le vieil RFC 2385, n'a au contraire rencontré aucun adepte. Même chose, hélas, pour TLS, pour le TCP-AO du RFC 5925 (ce dernier était pourtant recommandé par le RFC 6810), et pour IPsec. SSH, par contre, est une solution possible pour la moitié des implémentations. Bref, la sécurité des sessions RTR reste un sujet de préoccupation.

Et la gestion des différents codes d'erreur possible (section 6) ? À part JunOS, qui ignore les erreurs, tous ont répondu positivement.

Pour éviter de télécharger la liste des préfixes autorisés, avec leurs AS d'origine, à chaque changement, RTR a un système de mise à jour incrémentale. Nos programmes le mettent-ils en œuvre (section 7) ? Oui, sauf le serveur du RIPE-NCC et ceux des routeurs Cisco. Pour le concept de "*Session ID*", qui permet d'indiquer que le cache/validateur a redémarré, c'est mieux, seul le validateur du RIPE-NCC ne le gère pas (section 8).

La section 10 termine ce RFC avec des questions sur les tests d'interopérabilité qui ont été faits. On notera avec amusement que les vendeurs de solutions privatrices, comme Cisco ou Juniper, n'ont pas testé leurs produits entre eux, mais l'ont fait uniquement face aux logiciels libres.