RFC 6840 : Clarifications and Implementation Notes for DNS Security (DNSSEC)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 22 février 2013

Date de publication du RFC : Février 2013

https://www.bortzmeyer.org/6840.html

DNSSEC, normalisé notamment dans le RFC 4033 [1] et suivants, est une technologie très complexe et une erreur est vite arrivée lorsqu'on essaie de le mettre en œuvre. L'expérience de plusieurs programmeurs et de nombreux administrateurs système a permis de découvrir un certain nombre de faiblesses, voire de bogues, dans la spécification de DNSSEC et ce nouveau RFC rassemble des précisions et corrections. Leçon principale à en tirer : il n'y a pas que les logiciels qui ont des bogues, les normes aussi.

D'abord, des points qui étaient optionnels lors de leur introduction mais qu'on doit maintenant, au vu de l'évolution du monde, considérer comme faisant partie du cœur de DNSSEC, et sont donc obligatoires. C'est le cas de NSEC3, introduit par le RFC 5155. Étant donné que des zones très importantes comme `.com` et `.fr` sont signées avec NSEC3, un résolveur DNSSEC validant qui ne gérerait pas NSEC3 serait, en 2013, bien inutile... Le RFC demande donc que tous les résolveurs mettent en œuvre NSEC3. Notez que les anciens noms d'algorithmes DNSSEC <https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xml#dns-sec-alg-numbers-1> avaient une version NSEC **et** une version NSEC3 mais ce n'est plus le cas. l'algorithme 8, par exemple (RSA + SHA-256), permet d'utiliser NSEC ou NSEC3.

À propos de SHA, notre RFC considère aussi que la famille SHA-2 (RFC 5702) est désormais obligatoire (elle est utilisée pour la racine donc un résolveur qui ne la connait pas n'irait pas bien loin).

Après cet ajout de nouvelles techniques cryptographiques, un mot sur le passage à l'échelle (section 3). Notre RFC demande désormais fortement que les résolveurs DNSSEC validants aient un cache des erreurs de signature. Si une signature est invalide, réessayer immédiatement ne sert qu'à charger le

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc4033.txt

réseau inutilement : le problème ne va pas se résoudre seul. Un tel ré-essai ne peut mener qu'au « "*Rollover and die*" <https://www.bortzmeyer.org/rollover-and-die.html> ». Le RFC 4035, section 4.7, permettait un tel cache mais il est désormais fortement recommandé.

Bon, venons-en maintenant aux bogues et problèmes, en section 4. Le RFC suit dans cette section 4 à peu près un plan d'importance décroissante : au début, les bogues graves, pouvant mener à des failles de sécurité. À la fin, les améliorations non-critiques. Pour commencer, les preuves de non-existence. L'algorithme donné en section 5.4 du RFC 4035 est incomplet. Mis en œuvre littéralement, il permettrait de valider la non-existence d'un nom avec un NSEC ou un NSEC3 situé plus haut dans l'arbre, même en dehors de la zone. La nouvelle formulation impose au résolveur validant de ne pas chercher de NSEC (ou NSEC3) au delà des frontières de zone.

Autre sous-spécification : lorsque la requête est de type `ANY` (on demande tous les types connus du serveur), il est parfaitement légal de n'en renvoyer qu'une partie (un serveur ne faisant pas autorité n'a pas forcément en cache tous les enregistrements pour un nom donné, cf. section 6.2.2 du RFC 1034). Comment on valide cela ? Rien n'était dit dans le RFC 4035. La nouvelle règle est que, si un seul des ensembles d'enregistrements ("*RRset*") est invalide, toute la réponse doit être considérée comme invalide. Mais, si tous les ensembles sont valides, il ne faut pas pour autant que le résolveur croie qu'il a tout obtenu.

Il y a aussi deux autres erreurs de sécurité corrigées par notre RFC. Et le RFC continue avec la section 5, les problèmes d'interopérabilité. D'abord, encore un problème posé par une règle qui, dans le RFC 1034 (section 3.1), semblait bien innocente, celle comme quoi les comparaisons de noms de domaines sont insensibles à la casse. Or, DNSSEC nécessite souvent des comparaisons et des tris (par exemple pour ordonner les noms pour NSEC). Le RFC 3755 disait « "*Embedded domain names in RRSIG and NSEC RRs [donc, dans la partie droite des enregistrements] are not downcased for purposes of DNSSEC canonical form and ordering nor for equality comparison*" » mais le RFC 4034 disait le contraire « "*all uppercase US-ASCII letters in the DNS names contained within the RDATA are replaced by the corresponding lowercase US-ASCII letters*" ». Les deux formulations sont acceptables mais il faut évidemment que tout le monde utilise la même. Et que faisaient les résolveurs, face à cette contradiction ? Eh bien, leurs auteurs avaient tranché en ne mettant en œuvre aucun des deux RFC (le RFC ne dit pas comment ces auteurs étaient arrivés à cet accord) : les noms dans la partie droite d'un NSEC ne sont **pas** convertis en minuscule, ceux dans la partie droite d'un RRSIG le sont. Le RFC 6840 entérine cette pratique en en faisant la nouvelle règle.

Autre problème d'interopérabilité. Le RFC 4035 est clair sur ce que doit faire un résolveur qui ne connait aucun des algorithmes utilisés pour signer une zone (par exemple parce que l'auteur de la zone a choisi uniquement des algorithmes très récents comme ceux du RFC 6605) : dans ce cas, la zone est considérée comme non signée. Mais si un enregistrement DS est condensé avec un algorithme de condensation non connu du résolveur ? Le cas n'était pas prévu mais il l'est désormais : le résolveur doit faire comme s'il n'y avait pas de DS (donc considérer la zone comme non signée).

Un cas plus fréquent en pratique est celui d'un ensemble d'enregistrements signés plusieurs fois, par exemple avec des clés différentes (certains mécanismes de remplacement de clés fonctionnent ainsi), ou bien avec des algorithmes différents (par exemple lorsqu'on déploie un nouvel algorithme pas encore très connu et qu'on garde donc des signatures faites avec l'ancien algorithme). Si ces signatures donnent des résultats différents (mettons que l'une est valide et l'autre pas), que doit faire le résolveur ? On peut imaginer deux politiques possibles, imposer que toutes les signatures soient valides (politique "*AND*", la plus sûre) ou n'exiger qu'une seule signature valide pour être content (politique "*OR*", celle qui créera le moins de problèmes). La section 5.3.3 du RFC 4035 renvoyait le problème au résolveur en disant qu'il choisissait librement sa politique de sécurité. Notre RFC 6840 tranche plus nettement : il faut utiliser la politique "*OR*". Autrement, on ne pourrait pas introduire de nouveaux algorithmes cryptographiques. Et certains problèmes liés au moment exact où une donnée est introduite dans le cache

du résolveur mènerait à des échecs de validation (par exemple, remplacement d'une clé, l'ancienne clé n'est plus disponible mais des signatures faites avec elle sont toujours dans le cache). D'autre part, une politique "*AND*" faciliterait certaines attaques par déni de service : un méchant n'aurait qu'à insérer des signatures bidon pour empêcher la validation. Enfin, personnellement, je rajoute que DNSSEC est assez compliqué comme cela, qu'il y a déjà suffisamment de problèmes et qu'une politique laxiste est plus raisonnable à l'heure actuelle.

Autre cas où les RFC originels sur DNSSEC n'avaient pas assez pinaillé dans les détails et avaient omis de spécifier des cas un peu exotiques : que doit-on faire si le bit AD ("*Authentic Data*") est mis à 1 dans une **question** ? Désormais, cela a une sémantique précise : que le client DNS lit le bit AD, que sa valeur l'intéresse et que ce serait donc sympa de la part du serveur d'essayer de le déterminer. (C'est séparé du bit DO : le bit AD dans une requête n'implique pas qu'on veut recevoir toutes les données DNSSEC.)

Et le bit CD ("*Checking Disabled*") dans les requêtes ? Désormais, la règle est qu'il vaut toujours mieux le mettre à 1 lorsqu'on fait suivre une requête, afin d'augmenter les chances d'obtenir toutes les données disponibles chez le serveur en face. C'était une des deux décisions les plus contestées de ce nouveau RFC (l'autre étant le cas où plusieurs clés sont disponibles pour une zone). En pratique, cela n'a d'importance que si on a plusieurs résolveurs à la queue-leu-leu, par exemple avec la directive `forwarders` de BIND. Sans le dire clairement, le RFC 4035 semblait impliquer qu'il y ait au maximum un résolveur validant entre le "*stub resolver*" et les serveurs faisant autorité. Mais ce n'est pas toujours le cas. En suivant la nouvelle règle de ce RFC (toujours mettre le bit CD à 1 lorsqu'on fait suivre une requête), on simplifie : certes, il y a plusieurs validateurs sur le trajet mais un seul, le premier, fera la validation. Cela rendra le déboguage des problèmes plus facile. L'annexe B décrit plus en détail ce problème et ce choix. Elle présente des politiques alternatives, avec leurs avantages et leurs inconvénients. Notamment, mettre le bit CD à 1, dans certaines politiques, dépend de s'il était à 1 dans la requête originale.

Encore un cas vicieux ? Vous aimez cela ? Alors, demandez-vous ce que doit faire un résolveur validant avec DNSSEC lorsqu'il dispose de plusieurs clés de confiance pour valider une zone. Mettons (exemple purement théorique) qu'on ait la clé publique de la racine **et** qu'on ait installé celle de `gouv.fr` car on est un ministère et on veut pouvoir valider les domaines du gouvernement national sans dépendre de la racine. Quelle clé utiliser pour valider `culturecommunication.gouv.fr` ? Notre RFC ne tranche pas : cela reste une question de politique de sécurité locale (la question a été très disputée dans le groupe de travail et aucun consensus n'est apparu). Mais le RFC recommande que cela soit configurable, avec l'option par défaut la plus laxiste : « accepter n'importe quelle clé qui marche ». (Avec cette politique, la validation n'échoue que si toutes les clés échouent.) L'annexe C présente la liste des politiques possibles, avec leurs avantages et inconvénients :

— La clé la plus proche : collant à la nature arborescente et répartie du DNS, on choisit la clé la plus spécifique (ici, on choisira la clé de `gouv.fr` de préférence à celle de la racine, pour valider `culturecommunication.gouv.fr`). C'est la politique la plus raisonnable pour les cas où, comme dans mon exemple imaginaire, on a davantage confiance dans la clé située plus bas dans l'arbre. Mais elle nécessite une grande rigueur dans la gestion des clés des sous-zones. Si la clé de `gouv.fr` change, il faudra être sûr que tous les résolveurs gouvernementaux ont été mis à jour. (En pratique, je conseille de n'avoir au maximum que trois clés de confiance - incluant celle de la racine - et, de préférence, une seule - celle de la racine. Au delà, on risque vraiment d'avoir un jour ou l'autre une clé abandonnée.)

— N'importe quelle clé qui marche : c'est la politique recommandée par ce RFC, car c'est celle qui causera le moins d'erreurs de validation (les faux positfs étant une des plaies de la cryptographie). Son principal inconvénient est que la sécurité d'une zone est désormais celle de la clé la plus faible du lot.

— Choisir une clé en fonction de sa source : c'est plus compliqué à programmer mais l'idée est que toutes les clés ne se valent pas. Un validateur qui a DLV (RFC 5074), les DS trouvés dans le DNS et quelques clés gérées à la main peut décider, par exemple, qu'il fait plus confiance aux clés gérées à la main, quelle que soit leur place dans l'arborescence.

J'avoue n'avoir pas testé avec des validateurs DNSSEC existants quelle était leur politique...

Voilà, bon courage aux programmeurs, ils ont désormais un RFC de plus à garder sous le coude lorsqu'ils créent un logiciel DNSSEC... Une bonne partie des règles de ce RFC sont déjà largement mises en œuvre dans les validateurs existants comme BIND ou Unbound.