# RFC 8509 : A Root Key Trust Anchor Sentinel for DNSSEC

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 14 avril 2019

Date de publication du RFC : Décembre 2018

https://www.bortzmeyer.org/8509.html

---

On le sait, le premier changement de la clé DNSSEC de la racine du DNS s'est déroulé sans encombre le 11 octobre dernier. Ce qu'on sait moins est que ce changement a été précédé par des nombreuses études, pour tenter de prévoir les conséquences du changement. Une question importante était « combien de résolveurs DNS n'ont pas vu la nouvelle clé, depuis sa publication, et vont donc avoir des problèmes lorsque l'ancienne sera retirée du service ? » Ce RFC décrit une des techniques qui avaient été développées pour répondre à cette question, technique qui continuera à être utile pour les discussions actuellement en cours sur une éventuelle systématisation et accélération du rythme des changements de clé.

La question de cette systématisation a fait par exemple l'objet d'un débat à la dernière réunion IETF à Prague le 28 mars <https://datatracker.ietf.org/meeting/104/materials/agenda-104-ksk-00>. L'idée est de voir si on peut passer de changements exceptionnels et rares à des changements réguliers, banalisés. Pour cela, il faut avoir une idée de ce que voient les résolveurs DNS, du moins ceux qui valident avec DNSSEC, technique dont l'importance avait encore été démontré par les attaques récentes <https://www.bortzmeyer.org/dnspionage.html>. Mais comment savoir ce que voient les résolveurs et, notamment, quelle(s) clé(s) de départ de la validation ("*trust anchor*") ils utilisent ? La solution de la **sentinelle**, spécifiée dans ce nouveau RFC, peut permettre de répondre à cette question. L'idée est que les résolveurs reconnaitront une requête « spéciale », dont le nom commence par `root-key-sentinel-is-ta` ou `root-key-sentinel-not-ta`, nom qui sera suivi de l'identificateur ("*key tag*") de la clé (ta = "*Trust Anchor*"). La réponse du résolveur dépendra de s'il utilise cette clé ou pas comme point de départ de la validation. Un logiciel client pourra alors tester si son résolveur a bien vu le changement de clé en cours, et est prêt. L'utilisateur peut alors savoir si ce résolveur fonctionnera lors du changement. (Cela peut aussi aider l'administrateurice système mais celui-ci ou celle-là a d'autres moyens pour cela, comme d'afficher le fichier contenant la clé de départ de la validation. Par contre, les sentinelles sont utiles pour les chercheurs qui récoltent des données automatiquement, comme dans l'article de Huston cité à la fin.)

Ce mécanisme de sentinelle est complémentaire de celui du RFC 8145 [1], où le résolveur signale aux serveurs faisant autorité les clés qu'il utilise comme "*trust anchor*". Ici, la cible est le client du résolveur, pas les serveurs faisant autorité que contacte le résolveur. Ce mécanisme de sentinelle permet à tout utilisateur de savoir facilement la(es) clé(s) utilisée(s) par son résolveur DNS.

Petit rappel sur DNSSEC d'abord : comme le DNS, DNSSEC est arborescent. La validation part en général de la racine, et, via les enregistrements DS, arrive à la zone qu'on veut valider. Les clés sont identifiées par un "*key tag*" qui est un condensat de la clé. La clé qui est le point de départ de la validation se nomme le "*trust anchor*" et est stockée par le résolveur. Si on a la mauvaise clé, la validation échouera. Le "*trust anchor*" est géré manuellement par l'administrateur système en charge du résolveur, ou peut être mise à jour automatiquement en suivant la technique du RFC 5011. Aujourd'hui, le résolveur sur la machine où j'écris cet article utilise la "*trust anchor*" ayant le "*key tag*" 20326, la clé publique de la racine IANA (le résolveur est un Unbound) :

```
% cat /var/lib/unbound/root.key
...
. 172800 IN DNSKEY 257 3 8 AwEAAaz/tAm8y...1AkUTV74bU= ;{id = 20326 (ksk), size = 2048b} ;;state=2 [  VALID
```

Le `id = 20326` indique l'identificateur de la clé.

La section 2 expose le cœur du RFC. Un résolveur DNS, validant avec DNSSEC (RFC 4035) et qui met en œuvre ce RFC, doit reconnaitre comme spéciaux, les noms de domaine demandés qui commencent par `root-key-sentinel-is-ta-NNNNN` et `root-key-sentinel-not-ta-NNNNN` où NNNNN est un identificateur de clé. Voyons un exemple avec un domaine de test, dans lequel on trouve `root-key-sentinel-is` et `root-key-sentinel-not-ta-20326.ksk-test.net` (20326 est l'identificateur de la clé actuelle de la racine). Tout résolveur validant qui utilise la clé 20326 va pouvoir récupérer et valider le premier nom :

```
% dig root-key-sentinel-is-ta-20326.ksk-test.net
...
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23817
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 1
...
;; ANSWER SECTION:
root-key-sentinel-is-ta-20326.ksk-test.net. 30 IN A 204.194.23.4
```

Ici, le `ad` dans les résultats indique que l'enregistrement a bien été validé (AD = "*Authentic Data*"). Si le résolveur ne valide pas, on n'aura évidemment pas le `ad`. Maintenant, que se passe-t-il avec le second nom, celui avec `not-ta` ? Un résolveur validant, mais qui ne met pas en œuvre notre RFC 8509 va récupérer et valider ce nom :

```
% dig root-key-sentinel-not-ta-20326.ksk-test.net
...
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20409
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 1
...
;; ANSWER SECTION:
root-key-sentinel-not-ta-20326.ksk-test.net. 30 IN A 204.194.23.4
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc8145.txt`

En revanche, un résolveur validant **et** qui met en œuvre ce RFC 8509 (ici, le résolveur Knot <https://www.knot-resolver.cz/>) va renvoyer un SERVFAIL ("*Server Failure*") :

```
% dig root-key-sentinel-not-ta-20326.ksk-test.net
...
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 37396
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
```

On voit comment on peut utiliser le mécanisme de ce RFC pour découvrir si le résolveur utilise ou pas la clé 20326. Avec un client DNS, et une zone qui dispose des sentinelles :

- Si on peut récupérer `root-key-sentinel-not-ta-20326` et qu'on n'a pas le bit `ad`, c'est que le résolveur ne valide pas,
- Si on peut récupérer et valider `root-key-sentinel-not-ta-20326`, c'est que le résolveur valide mais ne gère pas le mécanisme des sentinelles,
- Si on peut récupérer et valider `root-key-sentinel-is-ta-20326` mais pas `root-key-sentinel-not-ta-20` c'est que le résolveur valide, gère le mécanisme des sentinelles, et utilise la clé 20326,
- Si on peut récupérer et valider `root-key-sentinel-not-ta-20326` mais pas `root-key-sentinel-is-ta-20` c'est que le résolveur valide, gère le mécanisme des sentinelles, et n'utilise pas la clé 20326. (Ce test ne marche pas forcément, cela dépend de comment est configurée la zone de test.)

L'annexe A de notre RFC détaille comment on utilise le mécanisme de sentinelle.

La beauté du mécanisme est qu'on peut l'utiliser sans client DNS, depuis une page Web. On crée une page qui essaie de charger trois images, une depuis `invalid.ZONEDETEST` (enregistrement mal signé), une depuis `root-key-sentinel-is-ta-20326.ZONEDETEST` et une depuis `root-key-sentinel-not-ta-20326.` On teste en JavaScript si les images se sont chargées :

- Si toutes se chargent, c'est que le résolveur de l'utilisateur ne valide pas,
- Si la première (`invalid`) ne charge pas mais que les deux autres chargent, c'est que le résolveur ne connait pas les sentinelles, on ne peut donc pas savoir quelle(s) clé(s) il utilise,
- Si la première (`invalid`) et la troisième (`root-key-sentinel-not-ta-20326`) ne se chargent pas mais que la deuxième (`root-key-sentinel-is-ta-20326`) se charge, c'est que le résolveur connait les sentinelles, et utilise la clé 20326.
- Si la première (`invalid`) et la deuxième (`root-key-sentinel-is-ta-20326`) ne se chargent pas mais que la troisième (`root-key-sentinel-not-ta-20326`) se charge, c'est que le résolveur connait les sentinelles, et n'utilise pas la clé 20326. Soit il utilise une autre racine que celle de l'ICANN, soit il a gardé une ancienne clé et aura des problèmes lors du remplacement.

Notez que le choix des préfixes avait été chaudement discuté à l'IETF. À un moment, l'accord s'était fait sur les préfixes `_is-ta` ou `_not-ta`, le tiret bas convenant bien à ces noms spéciaux. Mais comme ce même tiret bas rendait ces noms ilégaux comme noms de machine <https://www.bortzmeyer.org/host-vs-domain.html>, cela rendait difficile certains tests. Autant `_is-ta` ou `_not-ta` étaient utilisables depuis dig, autant on ne pouvait pas les tester depuis un navigateur Web, ce qui rendait difficile des tests semi-automatisés, sous formes d'images qu'on charge depuis une page Web et de JavaScript qui regarde le résultat. D'où les noms `root-key-sentinel-is-ta-NNNNN` et `root-key-sentinel-not-ta-NNNNN`. Ce n'est pas une solution satisfaisante que de prendre des noms ne commençant pas par un tiret bas, mais cela a semblé le meilleur compromis possible. Le nom est suffisamment long et alambiqué pour que le risque de collisions avec des noms existants soit faible.

Donc, que doit faire le résolveur quand on l'interroge pour un nom commençant par `root-key-sentinel-is-ta-NN` ou `root-key-sentinel-not-ta-NNNNN` ? Si et seulement si le type demandé est A (adresse IPv4) ou AAAA (adresse IPv6), et que la réponse est validée par DNSSEC, le résolveur doit extraire le "*key tag*" du nom et déterminer s'il corresponde à une des clés de départ de la validation pour la racine. (Les serveurs faisant autorité, eux, n'ont rien de particulier à faire, ils fonctionnent comme aujourd'hui.)

— Si le nom demandé commençait par `root-key-sentinel-is-ta` et que l'identificateur de clé est celui d'une "*trust anchor*", alors on renvoie la réponse originale,
— Si le nom demandé commençait par `root-key-sentinel-is-ta` et que l'identificateur n'est **pas** celui d'une "*trust anchor*", alors on renvoie le code d'erreur SERVFAIL,
— Si le nom demandé commençait par `root-key-sentinel-not-ta` et que l'identificateur est celui d'une "*trust anchor*", alors on renvoie le code d'erreur SERVFAIL,
— Si le nom demandé commençait par `root-key-sentinel-not-ta` et que l'identificateur n'est **pas** celui d'une "*trust anchor*", alors on renvoie la réponse originale.

Le principe est simple, les sections suivantes du RFC décrivent comment déduire l'état du résolveur avec ce principe. Par exemple, la section 3 décrit le cas où on n'a qu'un seul résolveur. Si on veut connaitre la situation de la clé 12345, et que les noms nécessaires sont présents dans le domaine xxx.example, que le domaine `broken.xxx.example` est cassé, question DNSSEC, on cherche à résoudre les noms `root-key-sentinel-is-ta-12345.xxx.example`, `root-key-sentinel-not-ta-12345.xxx.example` et `broken.xxx.example`. Pour un résolveur validant, et qui met en œuvre ce RFC, seule la première requête doit réussir. Si la troisième réussit, c'est que le résolveur ne valide pas. Si la deuxième réussit, c'est qu'il ne connait pas le système sentinelle de notre RFC 8509. (L'algorithme détaillé est dans la section 3 du RFC, il y a quelques cas vicieux.)

Notez bien qu'on n'a pas besoin d'un client DNS complet pour ces tests. Une résolution de nom en adresse normale suffit, ce qui permet de faire ces tests depuis un navigateur Web, ce qui était bien le but. Par exemple en HTML :

```
<ul>
 <li><a href="http://invalid.ksk-test.net/invalid.gif">"http://invalid.ksk-test.net/invalid.gif"</a></l
 <li><a href="http://root-key-sentinel-is-ta-20326.ksk-test.net/is-ta.gif">"http://root-key-sentinel-is
 <li><a href="http://root-key-sentinel-not-ta-20326.ksk-test.net/not-ta.gif">"http://root-key-sentinel-
</ul>
```

Et avec du JavaScript pour vérifier :

```
if (img_invalid.height === 0) {invalid = false;}
if (img_is_ta.height === 0) {is_ta = false;}
if (img_not_ta.height === 0) {not_ta = false;}

switch (true) {
  case invalid === true:
    result="You are not DNSSEC validating, and so will be fine!";
  break;
  case (is_ta === true && not_ta === true):
    result="You are using a legacy resolver (or updated resolvers, with some new and some old), we can
  break;
  case (not_ta === true):
     result="WARNING!: Your resolvers do not have the new KSK. Your Internet access will break!";
  break;
  case (is_ta === true):
     result="Congratulations, you have the new key. You will be fine.";
  break;
}
```

Si la machine a plusieurs résolveurs, le cas est plus compliqué par la bibliothèque qui fait la résolution de noms va typiquement passer au résolveur suivant en cas de réponse SERVFAIL. La section 4 décrit un algorithme qui peut permettre, sinon de déterminer la situation exacte de chacun des résolveurs utilisés, en tout cas de voir si une clé donnée a des chances de fonctionner avec cet ensemble de résolveurs.

Quels résolveurs ont ce mécanisme de sentinelle ? BIND l'a depuis la version 9.13, Knot <https://www.knot-resolver.cz/> a également ce mécanisme, activé par défaut, depuis la version 2.0 (et le documente <http://knot-resolver.readthedocs.io/en/stable/modules.html#sentinel-for-detectin Unbound en dispose depuis la version 1.7.1 et c'est activé par défaut. Parmi les résolveurs DNS publics, Cloudflare et Quad9 <https://www.bortzmeyer.org/quad9.html> ne gèrent apparemment pas les sentinelles de notre RFC 8509.

Côté client, on peut tester son résolveur avec dig, ou bien avec les services Web , , (le code derrière ce service est publié <https://github.com/paulehoffman/sentinel-testbed>, notez que les résultats sont présentés en utilisant des codes spécifiques au RFC, pas très lisibles) ou (utilise le domaine ksk-test.net). La lecture des sources de ces pages est recommandée.

On peut aussi regarder, parmi les sondes RIPE Atlas <https://atlas.ripe.net/>, lesquelles utilisent un résolveur avec sentinelles :

```
% blaeu-resolve --displayvalidation --type A --requested 100 --dnssec root-key-sentinel-not-ta-20326.ksk-test.ne
[ (Authentic Data flag)  204.194.23.4] : 30 occurrences
[ERROR: SERVFAIL] : 12 occurrences
[ (Authentic Data flag)   (TRUNCATED - EDNS buffer size was 4096 )  204.194.23.4] : 1 occurrences
[204.194.23.4] : 52 occurrences
Test #20692175 done at 2019-04-13T16:32:15Z

% blaeu-resolve --displayvalidation --type A --requested 100 --old_measurement 20692175 --dnssec root-key-sentin
[ERROR: SERVFAIL] : 1 occurrences
[ (Authentic Data flag)  204.194.23.4] : 38 occurrences
[204.194.23.4] : 56 occurrences
Test #20692176 done at 2019-04-13T16:33:56Z
```

Le premier test montre 52 sondes utilisant un résolveur non validant, 30 utilisant un validant sans sentinelles (ou bien utilisant une autre clé que 20326), et 12 utilisant un résolveur validant avec sentinelles. Le second test, effectué avec exactement les mêmes sondes, montre que les sondes utilisant un résolveur validant à sentinelles utilisent bien la clé 20326 (sauf une, qui a un SERVFAIL). Notez que les nombres ne coïncident pas parfaitement (30 + 12 - 1 ne fait pas 38), sans doute parce que certaines sondes ont plusieurs résolveurs DNS configurés, ce qui complique sérieusement l'analyse.

Un exemple d'utilisation de ce mécanisme de sentinelles figure dans cet article de Geoff Huston <https://www.potaroo.net/ispcol/2018-09/kskroll.html> et cet autre <https://www.potaroo.net/ispcol/2018-11/kskpm.html>.