RFC 5011 : Automated Updates of DNS Security (DNSSEC) Trust Anchors

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 23 octobre 2009. Dernière mise à jour le 2 juillet 2015

Date de publication du RFC : Septembre 2007

http://www.bortzmeyer.org/5011.html

La plaie de DNSSEC, comme celle de tous les systèmes de cryptographie, est la gestion des clés. Avec DNSSEC, la clé de signature des autres clés, la KSK ("*Key Signing Key*") est censée être remplacée ("*rolled over*") régulièrement. Si un résolveur a une KSK dans sa configuration, cela oblige l'administrateur du résolveur à effectuer le remplacement à la main, ce qui peut être contraignant. Notre RFC 5011 [1] propose une autre solution : le domaine signe la nouvelle KSK avec l'ancienne et le résolveur accepte alors automatiquement cette nouvelle clé.

Le résolveur peut ainsi construire une chaîne de confiance menant de la première clé pour laquelle il a été configuré, jusqu'à la clé actuelle. Cela implique que ledit résolveur ait un espace où écrire les clés successives qu'il a authentifié et décidé de garder.

La méthode de notre RFC présente des risques (cf. section 8.3) mais pas plus que la situation actuelle où les clés sont entièrement gérées à la main.

Ce RFC 5011 fournit donc un moyen de ne configurer la clé d'un domaine qu'une seule fois (cette configuration initiale reste manuelle), même si elle est remplacée par la suite. Le principe est simple (et le RFC est donc très court). La section 2 le décrit : quand une clé est configurée pour une zone du DNS (cette clé est dite un « "*trust anchor*" ») et qu'une nouvelle KSK apparait dans la zone, signée par l'ancienne, le résolveur accepte cette nouvelle KSK comme "*trust anchor*" et l'enregistre dans sa configuration. Une machine à états complète figure en section 4.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5011.txt

Le principal piège avec cette méthode est que, si une clé privée est volée, l'attaquant pourra continuer à générer une chaîne de confiance : le retrait de la clé compromise par le gérant de la zone ne suffira pas à annuler son usage. Il faut donc un mécanisme de révocation (section 2.1). Lorsqu'une KSK est publiée, signée mais que le nouveau bit `REVOKE` est mis, alors le résolveur doit retirer cette clé de sa configuration.

Un autre risque est que la clé privée soit volée sans que le gérant de la zone ne s'en rende compte immédiatement. Pour éviter que l'attaquant ne fasse accepter une nouvelle clé, le résolveur est supposé attendre lorsqu'une clé apparait (section 2.2). La durée d'attente est typiquement de 30 jours (section 2.4.1).

Le résolveur qui met en œuvre ce RFC 5011 doit périodiquement vérifier que la clé est toujours en service. La section 2.3 expose les règles à suivre pour cette vérification (il faut demander souvent pour détecter les nouvelles clés, sans attendre l'expiration du TTL mais pas trop pour ne pas surcharger les serveurs).

On l'a vu, ce RFC crée un nouveau booléen dans les enregistrement `DNSKEY`, pour indiquer la révocation. Le format est donc légèrement modifié, comme spécifié en section 3. C'est le bit n° 8 du champ "*DNSKEY Flags*" (voir section 2.1.1 du RFC 4034) qui servira à cet effet.

Si toutes les clés d'une zone sont révoquées, la zone devient non-validable (section 5), ce qui signifie qu'elle sera traitée comme si elle n'était pas signée du tout ("*insecure*" en terminologie DNSSEC).

Le bon fonctionnement de l'algorithme nécessite que la zone signée procède au remplacement des clés d'une manière particulière, décrite en section 6. Ainsi, le remplacement ("*rollover*") va nécessiter la révocation de l'ancienne clé.

Le logiciel BIND a introduit le système de ce RFC dans sa version 9.7. (Voir le fichier `README.rfc5011` dans le code source.) Cela se configure avec la directive `managed-keys`, qui ressemble beaucoup à `trusted-keys` (sauf l'ajout d'un mécanisme pour trouver la première clé, ici avec le paramètre `initial-key`), par exemple ainsi :

```
managed-keys {
    "example.com."     initial-key             257 3 5
                                "AwEAAeeGE5unuosN3c8tBcj1/q4TQEwzfNY0GK6kxMVZ
                                1wcTkypSExLCBPMS0wWkrA1n7t5hcM86VD94L8oEd9jn
                                HdjxreguOZYEBWkckajU0tBWwEPMoEwepknpB14la1wy
                                3xR95PMt9zWceiqaYOLEujFAqe6F3tQ14lP6FdFL9wyC
                                flV06K1ww+gQxYRDo6h+Wejguvpeg33KRzFtlwvbF3Aa
                                pH2GXCi4Ok2+PO2ckzfKoikIe9ZOXfrCbG9ml2iQrRNS
                                M4q3zGhuly4NrF/t9s9jakbWzd4PM1Q551XIEphRGyqc
                                bA2JTU3/mcUVKfgrH7nxaPz5DoUB7TKYyQgsTlc=";
                                // key id = 8779
};
```

Une fois que BIND démarre, il crée un journal pour une zone bidon, `managed-keys.bind` et commence le processus décrit dans le RFC :

```
23-Oct-2009 10:55:10.152 zone managed-keys.bind/IN/_meta: loaded serial 0
23-Oct-2009 10:55:10.169 zone managed-keys.bind/IN/_meta: Initializing automatic trust anchor management for
   DNSKEY ID 49678 is now trusted, waiving the normal 30-day waiting period.
```

On voit que `example.com` a déjà une nouvelle clé, 49678, signée avec l'ancienne (celle que j'ai configurée dans `manage-keys`) et que BIND commence la période d'attente. Voici, avec une zone différente, le contenu de `managed-keys.bind` au moment où une deuxième clé vient d'être publiée :

```
$ORIGIN .
$TTL 0 ; 0 seconds
@ IN SOA . . (
2          ; serial
0          ; refresh (0 seconds)
0          ; retry (0 seconds)
0          ; expire (0 seconds)
0          ; minimum (0 seconds)
)
KEYDATA 20150703040258 20150702160258 19700101000000 257 3 8 (
AwEAAaP3gGQ4db0tAiDEky0dcUNGeI1aTDYP5NFxzhbd
pD60ZhKLVV4KyxPmoSNUpq5Fv5M0iBwK1Tyswsyq/9sM
SoZ8zx8aT3ho1YnPsSqQeJfjTT1WsX6YZ5Kw6B2QkjRN
a6OMGZ96Kn8AI/slqsw+z8hY49Sn3baeo9iJxHPzloNc
2dQkW4aLqzNEYxnuoJsthCfGrPSAXlUjY9m3YKIaEWR5
WFYQk770fT+gGWLk/54Vp0sG+Lw75JZnwhDhixPFaToT
DNqbHQmkEylq1XJLO15uZ/+RZNRfTXZKO4fVR0tMEbMA
ITqRmyP8xLXY4RXbS4J32gnenQbzABX8sQmwO7s=
) ; KSK; alg = RSASHA256; key id = 55954
KEYDATA 20150703040258 20150702160258 19700101000000 257 3 8 (
AwEAAchb6LrHCdz9Yo55u1id/b+X1FqVDF66xNrhbgnV
+vtpiq7pDsT8KgzSijNuGs4GLGsMhVE/9H0wOtmVRUQq
Q50PHZsiqg8gqB6i5zLortjpaCLZS7Oke1xP+6LzVRgT
4c8NXlRBg3m/gDjzijBD0BMACjVGZNv0gReAg2OCr9dB
rweE6DnM6twG7D2NyuGjpWzKeJfNd3Hek39V9NGHuABG
kmYG16XCao37IWcP/s/57HuBom5U3SNfuzfVDppokatu
L6dXp9ktuuVXsESc/rUERU/GPleuNfRuPHFr3URmrRud
4DYbRWNVIsxqkSLrCldDjP1Hicf3S8NgVHJTSRE=
) ; KSK; alg = RSASHA256; key id = 24439
```

Pour Unbound, la configuration de ce RFC se fait en préfixant les directives par `auto-`. Ainsi, au lieu d'avoir une clé statique et qui ne bougera pas :

```
trust-anchor-file: "root.key"
```

On aura une clé modifiable :

```
auto-trust-anchor-file: "autokey/root.key"
```

Il faut bien sûr veiller à ce que le répertoire (ici `autokey/`) soit accessible à Unbound en écriture. Voici un fichier pour les mêmes clés que l'exemple BIND plus haut :

```
; autotrust trust anchor file
;;id: . 1
;;last_queried: 1435856265 ;;Thu Jul  2 16:57:45 2015
;;last_success: 1435856265 ;;Thu Jul  2 16:57:45 2015
;;next_probe_time: 1435899044 ;;Fri Jul  3 04:50:44 2015
;;query_failed: 0
;;query_interval: 43200
;;retry_time: 8640
. 86400 IN DNSKEY 257 3 8 AwEAAaP3gGQ4db0tAiDEky0dcUNGeI1aTDYP5NFxzhbdpD60ZhKLVV4KyxPmoSNUpq5Fv5M0iBwK1Tyswsyq/9
. 85667 IN DNSKEY 257 3 8 AwEAAchb6LrHCdz9Yo55u1id/b+X1FqVDF66xNrhbgnV+vtpiq7pDsT8KgzSijNuGs4GLGsMhVE/9H0wOtmVRU
```

Notez qu'Unbound, lui, écrit l'état des clés, `VALID` pour celle en cours d'utilisation et `ADDPEND` ("*Add, Pending*") pour celle qui la remplacera.

Un très bon article de test de notre RFC, avec OpenDNSSEC, BIND et Unbound, par Jan-Piet Mens <http://jpmens.net/2015/01/21/opendnssec-rfc-5011-bind-and-unbound/>.