RFC 4035 : Protocol Modifications for the DNS Security Extensions

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 5 juin 2008

Date de publication du RFC : Mars 2005

https://www.bortzmeyer.org/4035.html

L'ensemble connu sous le nom de DNSSEC vise à authentifier les réponses des serveurs DNS en signant cryptographiquement lesdites réponses. Cet ensemble est normalisé dans plusieurs RFC dont notre RFC 4035 [1], qui spécifie les changements dans le protocole. Ces modifications du protocole sont modérées mais indispensables pour l'authentification des données.

Ce RFC complète donc le RFC 4033 qui exposait les principes généraux de DNSSEC, ainsi que son cahier des charges, et du RFC 4034 qui décrivait les nouveaux types d'enregistrements DNS nécessaires, DS, DNSKEY, RRSIG et NSEC. Il met à jour l'ancien RFC qui couvrait tous les aspects de DNSSEC (RFC 2535). Il explique ce que doivent faire des serveurs et résolveurs DNSSEC pour permettre l'authentification. En effet, la simple utilisation des nouveaux enregistrements ne suffit pas à valider les réponses.

D'abord, la section 2 explique ce que doivent faire les programme de génération des zones (`dnssec-keygen` chez BIND) et de signature des zones (`dnssec-signzone` chez BIND). Elle parle donc du comportement « hors-ligne » du serveur, indépendamment des requêtes qu'il reçoit. La section 2.1 couvre la gestion des clés : leur partie publique doit être incluse dans la zone, dans des enregistrements DNSKEY, par exemple, ici dans `.bg` :

```
bg.   IN  DNSKEY  257 3 5 AwEAAcNsUEdXDwV5Sgr55kFzm0kCFNOL9iP8asHh1FKvDcxdJCe4mQJm l4kQ0C1CZyMwPIL3QWNP8yWcBLs2r
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc4035.txt`

La section 2.2 passe ensuite aux signatures. Elles doivent être mises dans des enregistrements RRSIG et il doit y en avoir un par "*RRset*" (ensemble des enregistrements de différents types pour un même nom de domaine). Cette section rappelle aussi qu'on ne signe que les enregistrements pour lesquels on fait **autorité**. Ainsi, les `NS` qui indiquent les serveurs de noms d'une zone, sont signés dans la zone fille, pas dans la zone parente qui s'en sert pour la délégation. Même chose pour la **colle <https://www.afnic.fr/observatoire-ressources/papier-expert/le-dns-ca-colle-ou-ca-ne-colle-pas/>** (adresses IP indispensables à la délégation), qui n'est pas signée.

Un des problèmes les plus difficiles de DNSSEC est la « preuve de non-existence ». Afin que les requêtes et réponses DNS puissent passer par des serveurs traditionnels (pour que le DNS continue à marcher, même sans validation), DNSSEC utilise des enregistrements DNS classiques. Prouver qu'un nom de domaine existe consiste simplement à produire une signature d'un enregistrement portant sur ce nom. Mais comment prouver qu'un nom de domaine n'existe pas, alors qu'il n'existe rien à signer ? DNSSEC résout ce problème par deux sortes d'enregistrement prouvant la non-existence, les NSEC, introduits dans le RFC 4034 et dont l'inclusion dans le fichier de zone est spécifiée en section 2.3 de notre RFC, et les `NSEC3` qui ont été introduits plus tard par le RFC 5155. Les NSEC qui, à l'époque de la publication de notre RFC 4035, étaient le seul mécanisme de preuve de non-existence, ont en effet un gros défaut : ils permettent à un client DNS de récupérer `<https://www.bortzmeyer.org/recuperer-zone-dns.html>` indirectement toute la zone DNS.

La section 2.3 dit donc que tout nom de domaine pour lequel un enregistrement est signé doit avoir un NSEC, par exemple, ici le MX de `laperouse.sources.org` est signé et un NSEC annonce que le nom suivant est `ludwigVII.sources.org` :

```
laperouse.sources.org.  86400   IN MX   10 aetius.bortzmeyer.org.
                        86400   RRSIG   MX 3 3 86400 20080802035206 (
                                        20080601035206 55957 sources.org.
                                        CH+eYPc36AXc9W9sEIaUFnOLdlodKXa+9yLQ
                                        SuXYgDbmEQMufy052L0= )
                        43200   NSEC    ludwigVII.sources.org. MX RRSIG NSEC
```

La section 2.4 explique le placement des DS, les enregistrements qui complètent les `NS` dans la zone parente, afin de permettre une délégation signée depuis une zone vers sa fille. Un DS référence les clés publiques de la zone fille. Les DS ne se trouvent donc pas dans la zone déléguée. Ils sont générés par le gérant de cette zone déléguée et transmis au gérant de la zone parente, qui les inclut dans son fichier de zone et les signe. (On pourrait aussi envoyer les DNSKEY au parent pour qu'il génère les DS lui-même mais le RFC 6781, section 4.3.2, recommande plutôt d'envoyer les DS.) Ainsi, si le gérant de `bortzmeyer.example` signe sa zone avec BIND, `dnssec-signzone` va lui générer un fichier `dsset-NOMDELAZONE.` qui contiendra les DS qu'il transmettra au gérant de `.example` (par exemple en EPP selon le RFC 4310 ou bien selon une procédure qui dépend du registre `<http://www.ripe.net/rs/reverse/dnssec/registry-procedure.html>`). Voici le contenu de ce fichier des DS :

```
sources.org.            IN DS 55957 3 1 A12F149F84B34E09501C32CC9F984FB9E1ED196A
sources.org.            IN DS 55957 3 2 DFE4A96D7FCC9B3F99990861AE3B915E832B699FC9628CE0DF7BE45D DE37DFB3
```

La section 2.5 parle des `CNAME`, ces enregistrements « alias » qui permettent à un nom de domaine de pointer vers un autre. Les `CNAME` doivent évidemment être signés, ce qui implique de leur adjoindre un RRSIG et notre RFC 4035 adoucit donc la vieille règle (RFC 1034) selon laquelle on ne pouvait pas mettre des données avec un `CNAME` :

```
openid.sources.org.     86400   IN CNAME munzer.bortzmeyer.org.
                        86400   RRSIG   CNAME 3 3 86400 20080802035206 (
                                        20080601035206 55957 sources.org.
                                        CFhUR1j+1b684LERarhf6bmIbWHvnox6/cZT
                                        txCiMpOS5rRHBcQQHYQ= )
                        43200   NSEC    sub.sources.org. CNAME RRSIG NSEC
```

De même, la section 2.6 modère l'ancienne règle qui disait que, dans la zone parente, seuls des `NS` pouvaient être présents pour une zone fille (désormais, les DS et les NSEC sont également possibles).

La section 3 s'attaque aux serveurs de noms qui veulent distribuer des données authentifiées. Ces serveurs se voient d'abors imposer de gérer l'extension EDNS0 (RFC 2671), notamment en raison de la plus grande taille de réponse qu'elle permet (les enregistrements DNSSEC sont typiquement assez grands). Mais c'est aussi parce que EDNS0 permet aux résolveurs DNS d'indiquer qu'ils acceptent DNSSEC, selon le mécanisme du *"DO bit"*, décrit dans le RFC 3225. Pour BIND, utilisé en résolveur, on met ce bit avec l'option `dnssec-validation yes;` dans le fichier de configuration. Pour dig, si je donne l'option `+dnssec`, dig va mettre ce *"DO bit"* (bit *"DNSSEC OK"*) dans la requête, comme on le voit ici avec wireshark et son option *"Export"* (tcpdump ne permet apparemment pas de l'afficher) :

```
    Additional records
        <Root>: type OPT
            Name: <Root>
            Type: OPT (EDNS0 option)
            UDP payload size: 4096
            Higher bits in extended RCODE: 0x0
            EDNS0 version: 0
            Z: 0x8000
                Bit 0 (DO bit): 1 (Accepts DNSSEC security RRs)
                Bits 1-15: 0x0 (reserved)
            Data length: 0
```

Outre ce bit dans les requêtes, DNSSEC prévoit deux bits dans les réponses, les CD (*"Checking Disabled"*) et AD (*"Authentic Data"*).

La section 3.1 traite d'un serveur faisant autorité pour une zone. Pour que BIND gère le DNSSEC dans ce cas, il faut mettre l'option `dnssec-enable yes;` dans son fichier de configuration. Un tel serveur qui reçoit une requête avec le bit DO doit inclure les enregistrements DNSSEC comme les RRSIG, et fournir les NSEC si le nom n'existe pas (ou s'il existe mais sans enregistrement du type demandé). Voici un exemple avec dig et BIND :

```
% dig +dnssec MX sources.org
...
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
# La réponse a été validée, "ad" est présent
...
;; ANSWER SECTION:
sources.org.            86379   IN      MX      10 uucp.bortzmeyer.org.
sources.org.            86379   IN      RRSIG   MX 3 2 86400 20080802035206 20080601035206 55957 sources.org. CE

# Et, si le nom n'existe pas :
% dig +dnssec MX nexistepas.sources.org
...
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 14178
# Le nom n'existe pas, NXDOMAIN
```

```
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 1
# Réponse validée grâce aux NSEC
...
;; AUTHORITY SECTION:
...
ludwigvii.sources.org.  10800   IN      NSEC    openid.sources.org. AAAA RRSIG NSEC
# Le NSEC dit qu'il n'existe rien entre ludwigvii.sources.org et openid.sources.org
```

L'inclusion des RRSIG (section 3.1.1) peut poser des problèmes de taille, car ils sont nettement plus gros que les enregistrements habituels. Le RFC détaille donc dans quels cas les envoyer. L'envoi des NSEC obét à des règles encore plus complexes, détaillées dans la section 3.1.3. Par exemple, s'il y a une réponse, mais qu'elle a été générée par un joker (le caractère * dans le fichier de zone), il faut envoyer les NSEC.

Après les serveurs faisant autorité de la section précédente, la section 3.2 parle des serveurs **récursifs**. Lorsque leur client indique son désir de faire du DNSSEC, via le bit DO, le serveur récursif doit leur envoyer les enregistrements DNSSEC et procéder à la validation, puis en indiquer le résultat via le bit AD (« ces données sont authentiques, je les ai vérifiées »). Pour BIND, c'est l'option `dnssec-validation yes;` qui met en route DNSSEC pour un serveur récursif.

Il y a tout un débat sur le meilleur endroit où effectuer la validation DNSSEC. Dans les applications pour qu'elles aient le plus d'informations possibles et puissent ajuster ainsi leurs messages à l'utilisateur ? Dans un résolveur situé sur la machine de l'utilisateur, la seule à laquelle il puisse vraiment faire confiance, puisque les autres machines et le réseau peuvent être sous le contrôle des méchants ? (Mais l'utilisateur de base ne va certainement pas installer et gérer BIND et il n'existe pas de serveur récursif et validateur « clés en main ».) Dans les résolveurs du FAI, qui sont déjà utilisés et configurés et qui sont gérés par des professionnels ? (Mais le FAI est parfois le premier à mentir, par exemple en dissimulant les réponses NXDOMAIN - ce domaine n'existe pas - derrière des fausses réponses menant à un serveur publicitaire qu'il contrôle.)

Ce débat n'aura pas de réponse consensuelle de si tôt (voir par exemple la section 4.9.3 du RFC). En attendant, notre RFC permet tous les modes de fonctionnement. Par exemple, si le résolveur ne veut pas que les serveurs auxquels il parle fassent de la validation, il peut leur demander avec le bit CD (section 3.2.2). S'il veut signaler à son client qu'il a validé (libre au client de le croire ou pas), il a le bit AD (section 3.2.3) pour cela.

La liste des tâches à accomplir par un résolveur continue en section 4, qui couvre tous les résolveurs, pas seulement les serveurs de noms récursifs de la section précédente. Eux aussi doivent gérer EDNS0 (section 4.1).

Si un résolveur valide les réponses avec DNSSEC, la section 4.3 donne les résultats auquel il peut parvenir :

— Sûres (”*Secure*”) : les réponses DNS ont été vérifiées avec DNSSEC et sont correctes. Le résolveur peut alors positionner le bit AD, comme précisé dans la section 4.9.3.
— Non sûres (”*Insecure*”) : en l'absence de clé cryptographique connue (voir le paragraphe suivant), les réponses DNS n'ont pas pu être vérifiées.
— Invalides (”*Bogus*”) : le résolveur avait une clé mais les signatures sont fausses. Cela peut être bien sûr parce qu'un attaquant a modifié les données mais c'est souvent aussi en raison d'une erreur des gérants de la zone signée, voire du résolveur ; signatures expirées (un mois par défaut, dans BIND), décalage de l'horloge (DNSSEC implique une synchronisation des horloges), modification maladroite d'un fichier, etc. Ce genre de problèmes est très fréquent en cryptographie pratique et sera certainement une source d'amusement sans fin pour les utilisateurs de DNSSEC. Si la réponse est invalide, un serveur récursif doit - section 5.5 - retourner le code `SERVFAIL` (”*Server failure*”).

— Incertaines (”*Indeterminate*”) : quelque chose a empêché le résolveur de se faire une opinion ferme.

Mais, justement, comment vérifier une signature ? Faut-il mettre dans le résolveur la clé de toutes les zones DNS du monde ? Évidemment pas. La section 4.4 parle des ”*trust anchors*” (« points de départ de la confiance »), ces clés (au moins une) qui sont mises en dur dans la configuration du résolveur. À partir d'une telle clé, on peut vérifier une zone, et toutes ces sous-zones signées, en suivant les enregistrements DS, où on trouvera les clés des sous-zones. Idéalement, si la racine du DNS était signée, un seul ”*trust anchor*” suffirait, la clé de la racine.

En pratique, la racine n'est pas actuellement signée. Il faut donc gérer à la main un ensemble de ”*trust anchors*” (voici par exemple le fichier que j'utilise avec BIND (en ligne sur `https://www.bortzmeyer.org/files/bind-trust-anchors`), qui est à jour le 2008-06-04). On retrouve ces clés sur des sites tels que `<https://www.ripe.net/projects/disi/keys/>` ou `<http://www.iis.se/domains/sednssec/publickey>`. La meilleure façon de gérer ces ”*trust anchors*” est décrite dans la section 5, qui décrit par exemple les vérifications de cohérence que devrait faire un résolveur (tester que la ”*trust anchor*” se retrouve bien dans la zone sous la forme d'un enregistrement DNSKEY). La section 5.1 parle notamment des « îlots de confiance » (”*islands of security*”), qui sont les zones signées sans que leur zone parente le soit (la très grande majorité, à l'heure actuelle). Par exemple, aujourd'hui, la zone `.org` n'est pas signée donc `sources.org`, qui l'est, est un « îlot de sécurité » dont les clés sont :

```
sources.org. IN DNSKEY 256 3 3 CKoOFAptkq5oqs8hlDf1gYDDpRPrjySc+wjBxLBbQtkXOrAUoBOrUa6I Yx83fYsA/i3BbhOXjS3K5/Jl
sources.org. IN DNSKEY 256 3 5 AwEAAduXW4l/heiryuKxgipsGR2mIDnZxotS1Cf6+hzYROeGqPUnO2SI aQVh8s89u2hUT6I8T0C0UGVA
```

La section 5.4 est consacrée aux NSEC, qui nécessitent des règles particulières pour leur validation.

Le reste du RFC est consacré à des exemples détaillés de zones signées, et de requêtes et réponses DNSSEC, avec leurs explications.

Si on veut tester un résolveur DNSSEC, il existe une zone prévue pour cela, `test.dnssec-tools.org`. Elle contient à peu près toutes les erreurs possibles. Par exemple, `futuredate-A.newzsk-ns.test.dnssec-tools.or` a une date de signature située dans le futur :

```
% dig A futuredate-A.newzsk-ns.test.dnssec-tools.org

; <<>> DiG 9.4.2 <<>> A futuredate-A.newzsk-ns.test.dnssec-tools.org
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 10278
```

Et, sur le serveur résolveur, le journal contiendra :

```
28-May-2008 14:07:59.910 dnssec: info: validating @0xa9c450: futuredate-A.newzsk-ns.test.dnssec-tools.org A: no
```

Voici quelques extraits des fichiers de configuration de BIND, pour faire du DNSSEC (on peut trouver des instructions plus détaillées dans le ”*DNSSEC HOWTO, a tutorial in disguise*” `<http://www.nlnetlabs.nl/dnssec_howto/>` ou dans ”*DNSSEC Resolver Test*” `<http://dnssec.vs.uni-due.de/>`) :

```
options {

        dnssec-enable yes; // Accepte les requêtes avec le bit DO et
        // renvoie les enregistrements DNSSEC

        dnssec-validation yes; // Valide les
 // enregistrements DNSSEC reçus (le bit DO est mis
 // systématiquement par les BIND récents)

};
```

Pour le débogage, il peut être prudent de mettre le résultats des traitements DNSSEC dans un fichier séparé :

```
 logging {
          channel dnssec_log {             // a DNSSEC log channel
                  file "/var/tmp/bindlog/dnssec.log" size 20m;
                  print-time yes;        // timestamp the entries
                  print-category yes;    // add category name to entries
                  print-severity yes;    // add severity level to entries
                  severity debug 7;      // print debug message <=7
                          // Cela va noter *beaucoup* de messages
                          // On peut aussi mettre "severity info;" qui
     // est nettement moins bavard.
          };

    category dnssec  { dnssec_log; };
};
```

Une fois que c'est fait, on peut tester le résultat avec des noms correctement signés comme `sigok.verteiltesys`
ou délibérement mal signés comme `sigfail.verteiltesysteme.net`.

Pour un serveur faisant autorité, il faut générer des clés par exemple :

```
% dnssec-keygen -a RSASHA1 -b 2048 -n ZONE sources.org
```

On peut aussi générer plusieurs clés de types différents et les inclure dans la zone, par exemple, pour une clé DSA, avec `dnssec-keygen -a DSA -b 1024 -n ZONE sources.org`. Notons que cette commande dépend du générateur aléatoire de la machine et elle consomme beaucoup d'entropie, se bloquant dès qu'il n'y en a pas assez de disponible. Paradoxalement, il vaut donc mieux l'exécuter sur une machine chargée.

Il faut ensuite signer la zone et penser à la resigner après chaque modification (ou bien avant son expiration, selon l'évenement qui a lieu en premier). Pour simplifer le processus, on peut utiliser make avec un `Makefile` de ce genre :

```
reload: sources.org.signed foobar.example.signed
        rndc reload
        # Ou nsdc rebuild && nsdc reload pour NSD

%.signed: %
        @echo Signing requires a lot of entropy in /dev/random, do not hesitate to load the machine...
        # 5356800 seconds = two months of validity
        dnssec-signzone -e +5356800 $^
```

Le serveur devra alors avoir dans son fichier de configuration (`named.conf` pour BIND, `nsd.zones` pour NSD), le fichier `sources.org.signed` (et non pas le fichier `sources.org` qu'on édite et qui ne contient pas les signatures).