# Sécurité du DNS et DNSSEC

Stéphane Bortzmeyer
AFNIC
Immeuble International – 78181 Saint-Quentin-en-Yvelines - France

## Résumé

*2008 a été l'année de la sécurité DNS, notamment avec l'annonce de la faille dite « Kaminsky » en juillet, faille qui rend beaucoup plus facile et rapide l'empoisonnement d'un serveur de noms. Comme la quasi-totalité des interactions sur Internet dépendent du DNS, cette faille a suscité beaucoup d'inquiétude. 2010 sera donc une année où on parlera souvent de la solution de sécurité DNSSEC, avec notamment la signature de la racine et de .FR.*

*En quoi consiste DNSSEC ? En une signature cryptographique des données distribuées par le DNS. Ces données pourront donc être validées par le résolveur DNS, quel que soit le chemin qu'elles ont suivi, si le résolveur connaît la clé publique de signature. DNSSEC est mis en œuvre dans pratiquement tous les serveurs en logiciel libre, qu'ils soient serveurs faisant autorité ou bien résolveurs.*

*DNSSEC peut protéger contre une large classe de problèmes, non seulement les attaques par empoisonnement mais aussi les serveurs menteurs comme les résolveurs DNS configurés par certains FAI pour rediriger les erreurs de frappe vers un site Web de publicité. En revanche, c'est une technologie délicate, qui nécessite beaucoup d'attention.*

## Mots clefs

DNS, DNSSEC, sécurité, cryptographie

# 1 Introduction

Le DNS (Domain Name System) est à la base de quasiment toutes les applications de l'Internet. Même si celles-ci, comme BitTorrent, utilisent des adresses IP, il y a presque toujours un moment, au moins au début de la « session » où des requêtes DNS sont faites. Ce système d'annuaire est un tel succès qu'il est désormais utilisé pour bien d'autres tâches que sa fonction originelle de traduction de noms en adresses IP.

Mais le DNS est vulnérable. Et, compte-tenu de son importance, cette vulnérabilité fait peser une menace sur tout l'Internet. Le DNS est vulnérable aux attaques par déni de service (qui ne sont pas traitées ici) mais surtout aux attaques par corruption des données, pour insérer des réponses trompeuses. Il est en effet relativement facile de glisser une réponse mensongère dans le processus de résolution DNS. Cette vulnérabilité, bien connue depuis longtemps, a pris une actualité particulière en juillet 2008 avec la publication des informations sur la « faille Kaminsky », qui était une méthode nouvelle, permettant d'exploiter beaucoup plus facilement et rapidement cette vulnérabilité.

Or, il existe depuis 2005 une méthode normalisée pour sécuriser le DNS contre ces tromperies : DNSSEC (pour « DNS Security Extensions »). Elle consiste en une **signature cryptographique** des données distribuées par le DNS. Elle a connu une brusque accélération médiatique avec l'annonce en octobre 2009 du calendrier de signature de la racine du DNS, prévue pour 2010.

Cet article explique les vulnérabilités du DNS, les principes de base de DNSSEC, son utilisation en pratique, l'état actuel de son déploiement. Il essaie également d'indiquer précisément ce que DNSSEC fait et ne fait pas, quels trous il bouche et quels trous il laisse ouvert. Comme toujours en sécurité, il n'y a pas de solution miracle, DNSSEC est un outil parmi d'autres, à savoir utiliser intelligemment.

# 2 Qu'est-ce qui ne va pas avec le DNS ?

## 2.1 Rappel

Je vais me permettre de commencer par un petit rappel DNS, essentiellement pour fixer le vocabulaire. Le DNS est un mécanisme de résolution décentralisé pour les **noms de domaines**[1]. Les noms de domaines sont attribués de manière

[1] Il y a d'autres mécanismes pour les mêmes noms, comme LDAP ou `/etc/hosts`.

arborescente, en commençant de la **racine**[2]. Sous celle-ci, se trouvent des domaines de tête ou TLD (*Top-Level Domain*), puis des domaines de second niveau et ainsi de suite. Un nom de domaine s'écrit en commençant par le domaine le plus profond, puis son **parent** (séparé par un point), puis le parent du parent et ainsi de suite jusqu'à la racine. Ainsi, `munzer.bortzmeyer.org` est un nom situé dans le TLD .ORG, le domaine de second niveau étant `bortzmeyer`. Ce caractère arborescent (ou hiérarchique) est essentiel dans le DNS et est largement utilisé par DNSSEC.

À ces noms sont associés des données. Dans le DNS, on parle d'**enregistrements**. Les enregistrements ont des formats variables, qui dépendent de leur **type**, noté par une courte chaîne de caractères. Les types les plus connus sont A (adresse IPv4), AAAA (adresse IPv6), MX (relais de courrier) et il y a des types utilisés par le DNS lui-même comme NS (serveurs de noms d'un domaine).

Pour la résolution de noms, il existe deux sortes de serveurs. Malheureusement, pour des raisons historiques, et aussi parce que le logiciel le plus utilisé, BIND[3], peut remplir les deux fonctions, ces deux sortes sont souvent confondues. Cela a des conséquences opérationnelles néfastes mais c'est surtout gênant pour la pédagogie. Je parlerai donc systématiquement de **serveurs faisant autorité**, pour ceux qui ont reçu les données à servir, et de **serveurs récursifs**, pour les **résolveurs**, qui font également presque toujours fonction de cache.

Un serveur faisant autorité (par exemple, BIND ou NSD[4]) est alimenté à partir d'un fichier (dit « fichier de zone ») ou d'une base de données. Ce fichier regroupe tous les noms d'une zone, une zone étant un ensemble connexe de domaines. Le serveur va ensuite servir les données à qui le demande. Les serveurs de l'AFNIC, qui servent .FR, sont ainsi des serveurs faisant autorité. Même chose pour ceux de la racine, les fameux « treize serveurs racine ».

Les serveurs récursifs, ou résolveurs, sont, eux chargés de servir une population plus restreinte, typiquement une université, un réseau local, ou bien les clients d'un même FAI. Ce sont eux que vous trouvez dans `/etc/resolv.conf` sur une machine Unix. Comme les serveurs faisant autorité, ils parlent le protocole DNS, et écoutent en UDP et TCP sur le port 53. Mais, contrairement à eux, ils ne connaissent pas de données au démarrage[5]. Ils apprendront ces données petit à petit en interrogeant les serveurs faisant autorité. Ils servent donc de relais pour les machines des utilisateurs finaux, parfois appelés (en anglais) *stub resolvers*.

Le serveur récursif qui veut résoudre `munzer.bortzmeyer.org` fonctionne en interrogeant d'abord la racine, qui lui indique les serveurs de .ORG, puis Afilias (gérant du .ORG), dont les serveurs lui indiquent à leur tour les serveurs de noms de `bortzmeyer.org`, puis enfin ces derniers, qui connaissent la réponse et la renvoient. Bien sûr, toutes ces informations sont gardées dans un cache, ce qui permet de ne pas consulter la racine et Afilias à chaque fois.

L'outil de déboguage le plus répandu, qui est au DNS ce que ping est à IP, se nomme dig. Il permet d'interroger un serveur (faisant autorité ou bien récursif) pour un nom et un type donné. Par exemple, on peut demander l'adresse IPv4 du serveur Web de l'Université d'Hiroshima, au Japon :

```
% dig A www.hiroshima-u.ac.jp

...
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39280
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 1

...
;; ANSWER SECTION:
www.hiroshima-u.ac.jp.      560     IN      A       133.41.4.55
```

Le *status* NOERROR signifie que tout s'est bien passé (NXDOMAIN indiquerait que le domaine n'existe pas). Les *flags* indiquent que le serveur interrogé était récursif (ra pour *Recursion Available*) et la réponse est `133.41.4.55`.

## 2.2 Vulnérabilités

Les vulnérabilités du DNS sont connues depuis longtemps. Contrairement à ce qu'ont raconté certains commentateurs lors de l'annonce de la faille Kaminsky en 2008, elles sont bien documentées, par exemple dans le RFC 3833, en 2005. Le fond du problème est que le DNS, pour diminuer la latence, et la charge du serveur, utilise essentiellement UDP. Ce protocole ne nécessite pas d'établissement de connexion avant d'échanger les données. Si l'adresse IP source est mensongère, ce qui est possible avec la plupart des FAI de la planète, il n'y a pas de moyen de le détecter[6]. Cela permet plusieurs attaques sur le DNS, notamment l'attaque par empoisonnement où le méchant va essayer de répondre à la place du serveur légitime. Le principe est

[2] Que les informaticiens, contrairement aux botanistes, situent en haut de l'arbre...

[3] `http://www.isc.org/software/bind`

[4] `http://www.nlnetlabs.nl/nsd/`

[5] Pour des raisons historiques, on rencontre souvent des serveurs qui font à la fois autorité et récursion. Mais c'est une mauvaise pratique.

[6] TCP n'a pas le même problème puisque l'attaquant qui utilise une adresse mensongère ne recevrait pas les réponses et ne pourrait donc pas compléter la connexion.

le suivant : au moment où un résolveur pose une question[7], l'attaquant répond avec son propre paquet UDP, contenant les données qu'il désire. Pour que la réponse soit acceptée par le résolveur, il faut reproduire certaines caractéristiques de la réponse attendue, notamment l'identificateur de la requête[8], l'adresse IP source et le port source. Cette attaque, et ses probabilités de réussite, est décrite en détail dans le RFC 5452.

La faille Kaminsky, annoncée en juillet 2008, augmente brutalement les chances de succès de cette attaque en utilisant non plus des noms réels (qui font que, si l'attaque rate, la bonne information se retrouve dans le cache, bloquant l'assaillant jusqu'à l'expiration du TTL) mais des noms inexistants. Kaminsky a transformé une attaque bien connue mais assez théorique et lointaine, en une vulnérabilité immédiate. Soudainement, en n'utilisant que des logiciels qu'on trouve facilement sur le réseau, n'importe qui pouvait empoisonner des résolveurs en quelques minutes, voire quelques secondes avec des perfectionnements ultérieurs.

La réaction immédiate, documentée après coup dans le RFC 5452, a été de rendre aléatoire la dernière variable qui ne l'était pas, le port source. Ce fut la SPR (*Source Port Randomization*), déployée massivement à l'été 2008[9]. La SPR était un peu la dernière cartouche. Les autres idées pour accroître l'entropie de la requête ne permettent plus de gagner grand'chose. Si la faille Kaminsky se perfectionne encore un peu, la seule ligne de défense restera DNSSEC.

# 3 Comment marche DNSSEC

DNSSEC repose sur quelques principes simples. Notamment, on ne modifie pas le protocole DNS. DNSSEC est une **extension** du DNS, pas un nouveau protocole. Ainsi, DNSSEC peut fonctionner au travers d'un cache non modifié. Un client non-DNSSEC peut interagir avec un serveur DNSSEC (et réciproquement).

Les autres principes sont :

- l'utilisation de la cryptographie asymétrique pour signer les données DNS,
- le fait que DNSSEC protège les **données** et non pas le **canal**,
- l'authentification de l'origine, pas forcément de la véracité.

Le second point signifie que DNSSEC protège de bout en bout. Les techniques de sécurité qui protègent le canal de communication (comme TLS pour SMTP ou, dans le cas du DNS, comme DNScurve[10]) sont en général plus simples à déployer mais ne protègent pas contre un intermédiaire non fiable. Ainsi, avec SMTP sur TLS, si le courrier est relayé par un serveur de messagerie qui triche, par exemple en modifiant le message, TLS n'empêchera rien. Au contraire, PGP est une technique de sécurisation de bout en bout : même si un serveur intermédiaire triche, PGP permettra de détecter la modification du message.

Contrairement à des techniques comme IPsec, DNSSEC permet donc d'authentifier le message, même s'il est passé par des relais douteux.

Enfin, le dernier point, le fait que DNSSEC garantisse l'authenticité de l'origine, mais pas forcément la « véracité » du message, est une limite qui existait déjà dans le DNS : celui-ci ne permet pas de dire qu'une information est « vraie ». Si l'administrateur système fait une faute de frappe et tape `19.0.2.1` comme adresse IP du serveur Web, au lieu de `192.0.2.1`, le DNS ne détecte pas l'erreur et sert la donnée « fausse ». De même, avec DNSSEC, si une base de données d'un bureau d'enregistrement n'est pas protégée contre l'injection SQL, l'attaquant pourra modifier cette base et DNSSEC n'y pourra rien, il intervient bien en aval.

## 3.1 Petit rappel de cryptographie

Cet article suppose connues les bases de la cryptographie asymétrique (celle où la clé comporte deux parties, une privée et une publique), ainsi que la notion de condensat (*hash*). Je rappelle que, pour signer, les données sont d'abord réduites à un condensat (dans le cas de DNSSEC, les algorithmes utilisés sont en général SHA-1 et la famille SHA-2), puis chiffrées avec la clé privée (dans le cas de DNSSEC, l'algorithme le plus fréquent est RSA).

## 3.2 La signature

Comme le cahier des charges de DNSSEC prévoyait de ne pas modifier le protocole DNS, les services cryptographiques sont fournis par des enregistrements DNS normaux. Les signatures sont ainsi transportés dans des enregistrements de type RRSIG. Un enregistrement RRSIG affirme l'authenticité d'un ensemble d'enregistrements d'un type donné. Par exemple, si je demande le MX de `iis.se` :

```
% dig +dnssec MX iis.se
```

J'obtiens l'enregistrement MX demandé ainsi que la signature :

```
iis.se.                 10      IN      MX      5 mx1.iis.se.
```

[7] Dans certains cas, l'attaquant peut choisir ce moment, par exemple en envoyant un courrier qui va nécessiter des tests faits avec le DNS.

[8] Sur 16 bits seulement, donc bien trop petit.

[9] On note que, à l'automne 2009, entre 15 et 20 % des résolveurs qui interrogent les serveurs de .FR n'utilisent encore qu'un seul port...

[10] `http://www.bortzmeyer.org/dnscurve.html`

```
iis.se.                 10      IN      RRSIG  MX 5 2 10 20091120152002
                    20091110152002 54842 iis.se. YwwdW67gA7...rE=
```

Cet enregistrement RRSIG comprend plusieurs champs :

- Les champs ordinaires d'un enregistrement DNS, nom de domaine, TTL, classe, type,
- le type couvert par la signature, ici, MX,
- les algorithmes utilisés, ici 5, pour[11] RSA/SHA-1 (un pour la signature et un pour la condensation),
- le nombre de labels dans le nom, ici 2 (`iis` et `se`),
- le TTL originel,
- la date d'expiration de la signature, ici le 20 novembre 2009,
- la date à partie de laquelle la signature est valide, ici le 10 novembre 2009,
- l'identité de la clé (un condensat de la clé), ici 54842,
- le nom signé,
- la signature elle-même, en Base64.

C'est ce dernier champ qui va permettre la validation par le résolveur.

## 3.3 Gestion des clés.

Pour valider une signature, il faut connaître la clé publique utilisée. Elle est elle-même distribuée dans le DNS, dans un enregistrement de type DNSKEY (+multi permet une présentation plus adaptée, avec l'identité de la clé) :

```
% dig +multi DNSKEY iis.se
...
iis.se.                 3594 INDNSKEY 256 3 5 (
                        BQEAAAABtZM6NXPiUetFT2dntmmMnHQj9MIlDOwxxmtm
                        ...
                        ) ; key id = 54842
```

Cet enregistrement contient :

- les paramètres DNS classiques, nom, TTL, classe, type,
- les options de la clé, ici 256, expliqué plus loin,
- le protocole, toujours 3 (il a existé d'autres versions de DNSSEC, jamais déployées),
- les algorithmes, ici 5 pour RSA/SHA-1,
- la clé elle-même, en Base64.

dig affiche en outre, grâce à l'option +multi, l'identité de la clé, ici 54842.

Une clé, dans DNSSEC, n'est pas un certificat. Elle n'est pas signée par une autorité (voir la section suivante sur le mécanisme de distribution et de validation des clés) et elle n'inclut pas de méta-données comme la date d'expiration (dans DNSSEC, les signatures expirent, mais pas les clés).

Le champ Options comporte seize bits. Les deux plus importants sont le septième (qui indique que la clé est une clé de zone utilisable par DNSSEC) et le quinzième. Ici, seul le septième est à 1, les options valent donc 256. Mais, en fait, la zone contient deux clés. Voici la seconde :

```
iis.se.                 3449 INDNSKEY 257 3 5 (
                        AwEAAcq5u+qe5VibnyvSnGU20panweAk2QxfIGVuVQhz
                        ...
                        ) ; key id = 18937
```

Il y a deux différences : l'enregistrement est plus long (car cette clé fait 2048 bits au lieu de 1024 pour la première) et le champ Options est à 257. Le quinzième bit est à 1 et il se nomme SEP pour *Secure Entry Point*. Il indique que cette clé peut être utilisée dans une configuration statique, par exemple dans le fichier de configuration d'un résolveur. La première clé, celle sans le SEP, ne doit pas être utilisée dans un tel but (en général, parce que sa stabilité n'est pas garantie).

Pourquoi deux clés ? En fait, cela n'est absolument pas indispensable dans DNSSEC. On peut tout faire avec une seule clé. Mais cela pose des problèmes pratiques. Les cryptographes recommandent de changer assez fréquemment (typiquement tous les trois ou quatre mois) la clé qui sert à signer la zone. Or, comme on le verra plus tard, la zone parente doit être prévenue de ce

[11] Voir `http://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml`

changement, ce qui n'est pas pratique. On a donc souvent deux clés, la ZSK (*Zone Signing Key*) et la KSK (*Key Signing Key*). La première sert à la signature de tous les enregistrements, n'a pas le bit SEP et change relativement souvent. La seconde ne sert qu'à signer la ZSK, a le bit SEP (et peut donc être distribuée par des moyens tels qu'une page Web, pour que des gens la récupèrent et la mettent dans leur fichier de configuration) et change rarement (disons tous les deux ou trois ans).

## 3.4 Délégation et validation

Une caractéristique essentielle du DNS est sa structure arborescente. Depuis la racine, des délégations successives mènent aux domaines de tête (TLD) puis aux domaines de second niveau, etc. DNSSEC s'appuie sur cette arborescence pour éviter les IGC traditionnelles de X.509. Avec DNSSEC, la validation des signatures suit celle des délégations.

### *3.4.1 DS*

Le mécanisme « normal » de délégation sécurisée avec DNSSEC est l'enregistrement de type DS (*Delegation Signer*). Stocké chez la parente de la zone signée, il contient un condensat cryptographique de la clé. En trouvant un DS (signé) chez le parent, le résolveur sait que la zone fille est signée. En trouvant la clé publique dans un enregistrement DNSKEY chez la fille, il peut le condenser et vérifier qu'il retrouve bien le DS. Cherchons le DS de `iis.se` :

```
% dig DS iis.se
iis.se.                 3600    IN      DS      18937 5 2 B5C422428D137FBF15E104...
iis.se.                 3600    IN      DS      18937 5 1 10DD1EFDC7841ABFDF...
```

Il y a deux DS (pour la même clé, la 18937), condensés avec des algorithmes différents (SHA-1 et SHA-2).

### *3.4.2 DLV*

Le problème des DS est qu'il faut que la zone parente soit signée et qu'on ait un moyen d'y ajouter des DS. À l'heure actuelle, la racine n'est pas signée, et aucun TLD ne peut donc y mettre des DS. Quand elle le sera, .COM ne le sera pas encore et les domaines en .COM ne pourront donc pas être sécurisés via un enregistrement DS. Même quand une zone parente est signée, cela ne veut pas dire qu'elle accepte déjà les DS (par exemple, .ORG ne les accepte pas encore). Enfin, pour les zones qui imposent l'utilisation d'un intermédiaire, le bureau d'enregistrement, il faut avoir un bureau d'enregistrement qui permettre la déclaration de DS. Bref, l'enregistrement DS a pour force d'utiliser la hiérarchie existante du DNS et pour faiblesse de dépendre de cette hiérarchie.

Une solution possible est donc de contourner la hiérachie en faisant appel à un serveur DLV. DLV (*DNSSEC Look-aside Validation)* permet d'avoir une délégation des signatures qui ne passe pas par l'arborescence normale. Par exemple, si je signe `sources.org` et que mon bureau d'enregistrement ne peut pas ou ne veut pas transmettre le DS à Afilias, je peux le mettre dans `dlv.isc.org` (l'ISC gère en effet le plus gros registre DLV actuel). Un résolveur configuré pour tester dans `dlv.isc.org` pourra alors trouver mon DS.

En fait, on n'utilise pas d'enregistrement DS mais un enregistrement DLV, qui a exactement le même format. Voici celui de `sources.org` :

```
% dig DLV sources.org.dlv.isc.org
sources.org.dlv.isc.org. 3600      IN      DLV     14347 3 1 31FF6986A07DAC36...
sources.org.dlv.isc.org. 3600      IN      DLV     14347 3 2 0D5D5B209264B90107...
```

## 3.5 Signer le vide

### *3.5.1 Le problème*

DNSSEC repose sur la signature des enregistrements transmis par le serveur. Mais s'il n'y a pas d'enregistrements à signer, par exemple parce que le nom n'existe pas ? Signer la réponse nécessiterait un changement du protocole, qui ferait que, par exemple, les caches intermédiaires ne pourraient plus être utilisés sans mise à jour. DNSSEC a donc choisi une autre approche, reposant sur des enregistrements nouveaux, qui indiquent la non-existence. Ces enregistrements peuvent alors être signés par les mécanismes normaux de DNSSEC.

### *3.5.2 NSEC*

Le premier de ces enregistrements nouveaux se nomme NSEC pour *Next Secure*. Son principe est d'indiquer l'enregistrement suivant. Ainsi, si `ns2.nic.example` existe, ainsi que `www.nic.example`, le signeur de la zone générera un enregistrement NSEC qui indique qu'il n'y a rien entre ces deux noms (l'ordre utilisé est l'ordre lexicographique) :

```
ns2.nic.example.    7200    IN A    198.51.100.214
                    7200    NSEC    www.nic.example. A RRSIG NSEC
```

Plus précisement, l'enregistrement NSEC se lit « Il n'existe rien entre ns2.nic.example et www.nic.example, et ns2.nic.example n'a que des enregistrements de type A, RRSIG et NSEC ».

C'est cet enregistrement qui sera renvoyé au client DNS qui demanderait, par exemple, des informations sur ftp.nic.example, nom inexistant et situé lexicographiquement entre ces deux-là. Ainsi, un validateur DNSSEC devra, d'abord vérifier la signature du NSEC, puis calculer que le nom qu'il avait demandé est bien dans la plage que couvre le NSEC.

Les NSEC sont simples et résolvent bien le problème. Mais ils présentent un risque d'énumération. En demandant un nom au hasard, un attaquant obtient le nom suivant le sien, puis, en demandant ce nom auquel il rajoute un caractère, il récupère le nom suivant et ainsi de suite. Il peut donc énumérer la zone, obtenir tout son contenu. Pour plusieurs administrateurs de zone, par exemple pour la plupart de ceux des TLD européens, c'est une indiscrétion inacceptable.

### *3.5.3 NSEC3*

NSEC3, normalisé dans le RFC 5155, résout le problème de l'énumération de manière élégante. Au lieu d'indiquer les noms de domaine qui encadrent le nom inexistant, NSEC3 indique les condensats qui encadrent le condensat du nom inexistant. Comme on ne peut pas remonter d'un condensat au nom originel, il ne distribue ainsi pas plus d'information que nécessaire. En contrepartie, il oblige le serveur faisant autorité à effectuer un petit calcul cryptographique à chaque requête pour un nom inexistant. Et il est certainement complexe, car il y a de nombreux cas à prendre en compte.

Néanmoins, vu le problème de l'énumération, NSEC3 est utilisé par toutes les zones qui ne veulent pas que leur contenu soit connu, comme .ORG ou .COM.BR.

Voici un exemple de zone signée avec NSEC3 :

```
QGI7R9ORDRTCCHG37VR2J667TATVI1HU.example. 7200 IN NSEC3 1 0 4
                    DEADBEEF U2M7KDE1UAP8B22P9RIFMBBGJI7GFUKT AAAA RRSIG
QGI7R9ORDRTCCHG37VR2J667TATVI1HU.example.  7200  RRSIG  NSEC3 7 2
                          7200 20091214073024 (
                                    20091114073024 64479 example.
                                    eHowPzchbLHohe/6K+zo9XGmoB80YdsK4MkR ...
```

QGI...1HU est le condensat de www.nic.example[12]. Le condensat suivant est U2M...UKT, celui de test.example. Cet enregistrement NSEC3, traduit en français, dit « Il n'existe pas de nom dont le condensat soit entre QGI...1HU et U2M...UKT ».

Si un résolveur interroge le serveur sur le nom ftp.nic.example, le serveur pourra, en prime du code NXDOMAIN, envoyer ce NSEC3. Le résolveur calcule le condensat de ftp.nic.example, trouve SU5...J8O, qui est effectivement situé entre les deux ci-dessus. Donc, en effet, ftp.nic.example n'existe pas.

L'enregistrement NSEC3 est lui-même signé, pour pouvoir vérifier son authenticité. C'est le rôle du deuxième enregistrement, le RRSIG.

### *3.5.4 Le protocole*

DNSSEC a nécessité quelques petits ajouts au protocole DNS (qui n'empêchent pas l'interopérabilité avec les « vieux » logiciels). Ainsi, DNSSEC nécessite l'emploi de EDNS0 (RFC 2671) pour pouvoir passer un nouveau booléen, nommé DO, et qui indique que le résolveur comprend DNSSEC. D'autre part, DNSSEC a réservé deux booléens qui étaient libres dans les *flags* DNS, baptisés CD et AD. Le premier (*Checking Disabled*), envoyé dans une question, indique d'un client demande au résolveur de ne pas faire de validation. Le second (*Authentic Data*), figurant dans une réponse, indique que le résolveur a validé avec DNSSEC et trouvé la réponse correcte.

# 4 Ce que fait (et ne fait pas) DNSSEC

Arrivé à ce stade, je préfère résumer ce que fournit exactement DNSSEC comme service. Il assure la vérification de l'origine des données. Avec DNSSEC, si l'administrateur de wikipedia.fr a décidé de publier deux adresse IP pour www.wikipedia.fr, DNSSEC vous garantit la possibilité de vérifier que votre résolveur n'a pas été trompé par un intermédiaire. Les données sont bien comme à l'origine.

On comprend mieux les services que rend DNSSEC quand on voit ceux qu'il ne rend pas. DNSSEC, comme toute solution de sécurité, n'est pas parfait, il ne règle pas tout (et ne fait pas le café). D'abord, il y a les services qui ne figurent pas dans son cahier des charges. DNSSEC ne fournit pas de service de confidentialité. Si on veut empêcher un méchant de voir les requêtes DNS qu'on fait, il faut tunneler, par exemple dans IPsec.

[12] On ne peut pas remonter du condensat au nom qui a été condensé. J'ai donc trouvé cette information par force brute, en testant toute la zone.

Ensuite, DNSSEC ne protège pas l'intégralité de la session utilisateur. Si, par exemple, une attaque BGP détourne le trafic destiné à vos adresses IP, le fait que vos clients obtiennent la bonne adresse IP n'empêchera rien.

Surtout, DNSSEC assure la vérification de l'origine, pas la « véracité » des données. Si les données rentrées par l'ingénieur système dans la zone sont fausses (par exemple suite à une négligence), DNSSEC ne changera rien. Des attaques comme celle contre le bureau d'enregistrement de `google.ma`, en 2009 ou comme celle contre le registre de .PR, la même année, attaques toutes les deux lancées par injection SQL[13] ne pouvaient pas être détectées par DNSSEC. Même chose pour la grande panne qui a affecté .SE, en octobre 2009[14], panne provoquée par une bogue dans le générateur de zone, n'a pas été détectée par DNSSEC.

Enfin, comme toute technique de sécurité, DNSSEC amène ses propres risques, notamment celui de faux positifs. Si une erreur survient (la plus courante, à l'heure actuelle, est l'expiration d'une signature), DNSSEC peut déclarer invalide des domaines qui étaient corrects précédemment.

Bref, DNSSEC ne résout pas tous les problèmes. C'est un outil dans la boîte à outils du responsable de la sécurité du réseau, pas une baguette magique.

# 5 Je veux faire du DNSSEC

## 5.1 Signer ses domaines

Je ne vais pas présenter le problème dans l'ordre chronologique. Normalement, il faut d'abord générer ses clés (après avoir pris des précautions pour leur conservation en sécurité), puis signer, puis servir les zones signées. Mais il est plus simple de commencer par la fin.

### *5.1.1 Servir une zone signée*

Supposons donc que la zone signée existe. Que doit savoir faire le serveur de noms ? DNSSEC a été conçu pour que, si une zone est signée, toutes les opérations de cryptographie (qui sont typiquement assez coûteuses) puissent être faites hors-ligne. Au moment où la requête DNS arrive, le serveur de noms n'a pas besoin de faire de cryptographie (il existe une petite exception, détaillée plus loin). Cela limite donc la charge des serveurs faisant autorité et donc les ressources matérielles nécessaires. En revanche, les réponses DNSSEC sont typiquement de bien plus grande taille (on passe d'environ 100 octets à plus de 500, voire plus de 2000) et la charge réseau va donc augmenter.

Le serveur doit comprendre DNSSEC, pour pouvoir servir les NSEC et les NSEC3 intelligemment. Il faut donc vérifier que son serveur gère la partie de DNSSEC qui nous intéresse. Parfois, il faut activer le fonctionnement DNSSEC (dans BIND, c'est la directive `dnssec-enable yes;`) Par exemple, pour BIND, NSEC3 n'est disponible que depuis la version 9.6.

Le seul cas où le serveur faisant autorité doit faire un peu de cryptographie est avec NSEC3. Lorsqu'une requête pour un nom non-existant arrive, il faut condenser ce nom. Par exemple, les algorithmes utilisant pour cela SHA-2 ne sont disponibles dans BIND qu'à partir de la version 9.7.

### *5.1.2 Signer une zone*

Il existe à l'heure actuelle plusieurs logiciels de signature DNSSEC pour des fichiers de zone à la syntaxe standard. On peut aussi signer dynamiquement (BIND, par exemple, sait le faire, ce qui est nécessaire pour les mises à jour DNS dynamiques). J'utiliserai deux outils ici, dnssec-signzone, livré avec BIND et ldns-signzone, livré avec la bibliothèque de développement DNS ldns[15].

Pour signer une zone avec ldns, la commande est (en supposant que la clé privée soit dans le fichier `Kexample.+005+06612`) :

```
% ldns-signzone example.zone Kexample.+005+06612
```

On trouve alors un fichier de zone `example.zone.signed` qui contient les signatures. Ce fichier peut être chargé par le serveur de noms.

Si on a séparé ZSK et KSK, il faut les indiquer toutes les deux sur la ligne de commande. ldns-signzone saura s'y retrouver.

Avec le signeur de BIND, il faut (jusqu'à la version 9.6 incluse) ajouter les clés manuellement dans le fichier (ici, une ZSK et une KSK) :

```
% cat example.zone \
        Kexample.+005+26011.key Kexample.+005+53697.key > example+key.zone
```

On peut ensuite signer :

```
% dnssec-signzone   -o example example+key.zone
```

Et les performances ? Voici le résultat sur le fichier de zone de .FR, 1,55 millions de noms, 155 méga-octets :

[13] http://www.bortzmeyer.org/attaques-registres-noms.html

[14] http://royal.pingdom.com/2009/10/13/sweden%25E2%2580%2599s-internet-broken-by-dns-mistake/

[15] http://www.nlnetlabs.nl/projects/ldns/

```
dnssec-signzone -o fr fr.db  748.78s user 22.27s system 305% cpu 4:12.41 total
```

soit quatre minutes seulement (la machine était un quadri-processeur). Le fichier à servir est passé à 603 méga-octets.

Avec ldns, on obtient à peu près le même temps CPU (mais il ne sait pas utiliser les multi-processeurs donc le temps écoulé est plus long).

Si on signe avec NSEC3, le temps de calcul est à peu près le même. Par contre, NSEC3 offre une option nouvelle, l'*opt-out*, qui permet de ne pas signer tous les enregistrements mais seulement ceux qui font autorité. Dans une zone « terminale », une zone ordinaire qui n'a pas de délégation, cela ne change rien. Mais pour une zone composée presque uniquement de délégations, dont la plupart ne sont pas signées, comme c'est le cas de .FR, le gain est spectaculaire. On signe désormais en bien moins d'une minute, et le fichier de zone ne grandit pratiquement plus[16].

À noter que la signature consomme surtout du temps CPU et la génération de clés surtout de l'entropie, car elle a besoin de beaucoup de données réellement aléatoires. Par défaut, sur Linux, BIND et ldns utilisent `/dev/random`, qui est efficace mais peut bloquer si la machine n'a pas accumulé assez d'entropie (par exemple parce qu'elle vient de démarrer). Paradoxalement, on accélère souvent le processus en chargeant la machine (par exemple avec find) car cela fournit de l'entropie à `/dev/random`.

Pour les signatures, coûteuses en temps CPU, BIND peut utiliser des accélérateurs matériels. Notez toutefois que la configuration exacte pour que ça marche est hautement non triviale, la spécification d'interface PKCS #11 étant complexe et mal respectée.

### *5.1.3 Gérer ses clés*

La gestion des clés est souvent présentée, à juste titre, comme l'aspect le plus difficile de la cryptographie. Gérer ses clés, c'est les stocker de manière sûre (sûre à la fois contre la copie non autorisée, et contre la perte) et les retrouver lorsqu'on en a besoin. Ces exigences sont souvent antagonistes. Par exemple, si on duplique la clé privée sur plusieurs sites physiques, on augmente ses chances de la récupérer, même en cas d'accident comme un incendie. Mais on augmente aussi les risques de copie non autorisée, puisque le voleur peut désormais choisir parmi plusieurs sites pour opérer. Et il faut bien sortir la clé privée de son coffre-fort pour signer...

Il existe plusieurs approches de la gestion de clés, selon qu'on est « cool » ou ultra-militarisé. Notons que la plupart de ces approches n'ont rien de spécifique à DNSSEC. Ce sont les mêmes lorsqu'on doit gérer des clés PGP ou des clés DKIM.

La méthode la plus classique et la plus habituelle pour les ingénieurs système Unix est de mettre la clé dans un répertoire, dont le propriétaire et le groupe sont bien choisis, et qui est en mode 0700 (seul le propriétaire peut faire quelque chose, les autres ont zéro droit). Il est recommandé que la machine qui porte les clés soit soigneusement sécurisée (pas de compte d'étudiants, ou de scripts en PHP dessus). Ce système a l'avantage d'être simple à mettre en œuvre. Signer est facile puisque la clé est toujours accessible via le réseau. Sa sécurité est celle de la machine et nous savons tous que les failles de sécurité surgissent vite. Toutefois, si l'alternative est entre l'actuelle absence de sécurité du DNS et une sécurité modérée apportée par ce système, il est intéressant.

Une méthode un peu plus sûre est de mettre la clé sur un support physique (clé USB, par exemple) enfermé dans un coffre-fort. Pour la signature, on sort la clé du coffre. Si on est vraiment prudent, on signe sur une machine déconnectée du réseau. Ce système est nettement moins pratique mais a l'avantage de s'appuyer sur les matériels et procédures de sécurité physique qui existent dans certaines organisations.

Enfin, la solution de luxe est de stocker les clés dans un matériel dédié, le HSM (*Hardware Security Module*). Le HSM est une carte ou une machine spécialisée, résistante à l'agression physique (sur les modèles les plus perfectionnés, la clé privée est automatiquement effacée en cas de tentative d'ouverture agressive). La clé ne sort pas du HSM, qui assure les fonctions de signature lui-même. Le HSM peut donc être laissé dans la salle machine, connecté au réseau. On trouve de tels systèmes entre 1 000 et 10 000 euros.

Dans tous les cas, il faudra définir des procédures de gestion des clés, par exemple de qui peut ouvrir le coffre, qui doit être présent, etc. Même chose pour le remplacement des clés. Même si on ne partage pas l'inquiétude de certains cryptographes quant à la nécessité de changements fréquents, il faut bien voir que des procédures qui ne sont que rarement exécutées ont peu de chances de l'être sans problèmes, s'il faut les faire en urgence. Ainsi, il vaut mieux changer les clés sans que ça soit nécessaire, pour que cela devienne une opération de routine.

Pour générer des clés avec ldns, on procède ainsi pour une ZSK :

```
% ldns-keygen -a RSASHA1 example
Kexample.+005+06612
```

`Kexample.+005+06612.key` contient la clé publique (au format des fichiers de zone), Un `Kexample.+005+06612.ds` a été créé et contient le DS. Le fichier `Kexample.+005+06612.private` doit être soigneusement protégé, il contient la clé privée.

Pour une KSK, on rajoutera l'option `-k`.

[16].FR sera très probablement signé avec cette option.

Si on utilise BIND pour fabriquer les clés (notez que les commandes DNSSEC de BIND seront très simplifiées à partir de la version 9.7 ; ici, on a utilisé le 9.5), la commande est (il n'y a pas de taille par défaut, il faut en spécifier une) :

```
% dnssec-keygen -a RSASHA1 -b 1024 example
Kexample.+005+53697
```

Et, pour avoir une KSK, on utilise l'argument `-f KSK`. Dans les deux cas, on trouve également partie privée et partie publique de la clé dans deux fichiers différents.

Connaître ces commandes est utile si on fait tout à la main ou bien si on écrit soi-même les `Makefile` qui vont gérer les opérations, mais on peut aussi utiliser un logiciel qui automatise une grande partie de ces tâches. C'est le cas, par exemple, d'OpenDNSSEC[17]. On configure ce dernier avec la politique qu'on souhaite suivre (par exemple, changement de ZSK tous les quatre mois, de KSK tous les deux ans, etc) et il génère automatiquement les clés nécessaires[18], il met les clés dans le fichier de zone, il signe, tout cela sans intervention humaine. OpenDNSSEC est actuellement en version beta mais est très prometteur.

## 5.2 Valider les signatures

### *5.2.1 Activer la validation*

Le résolveur validateur, lui, va faire des calculs cryptographiques à chaque fois qu'il validera une signature. Il faut donc prévoir une machine ayant de bonnes capacités de calcul.

Mais le principal risque lorsqu'on active la validation sur ses résolveurs est celui des faux positifs. En effet, si un problème survient (mauvaise signature sur la zone faisant autorité, signature expirée, enregistrement DS obsolète, etc), les données seront vues comme invalides et rejetées. On perd donc de la robustesse du DNS, au profit de la sécurité. L'expérience de la cryptographie sur Internet montre que les erreurs de validation sont bien plus souvent dues à des erreurs commises par les émetteurs légitimes qu'à des attaques. Ainsi, en septembre 2009, le registre de .PR a procédé à un changement de sa clé et a retiré l'ancienne clé seulement deux jours après. Le registre DLV de l'ISC ne se synchronisant que toutes les semaines, tous les noms en .PR disparaissaient[19], pour les utilisateurs de ce registre (la grande majorité des résolveurs validants). Avant DNSSEC, le DNS était la statue d'un chat : une fois finie, elle restait sur l'étagère et n'avait pas besoin d'entretien. Avec DNSSEC, le DNS devient un chat vivant, il faut s'en occuper, changer sa litière, etc.

Il convient donc de bien peser le pour et le contre avant d'activer la validation DNSSEC.

Une fois qu'on est décidé, comment faire ? Avec Unbound[20], dès qu'on a configuré au moins une clé de confiance, la validation sera faite. Avec BIND, il faut l'option `dnssec-validation yes;`. Dans les deux cas, si le domaine doit être signé mais que la signature des données n'est pas correcte, aucune donnée ne sera renvoyée au client DNS, uniquement un code SERVFAIL, *Server Failure*.

Qu'est-ce qu'une clé de confiance ? La validation DNSSEC nécessite un point de départ, une clé publique qui servira à valider un domaine et, par le jeu des délégations, tous les domaines en dessous. Dans le cas le plus simple, la racine du DNS est signée et tous les domaines de tête (les TLD) également. On n'a alors besoin que d'une seule clé de confiance, celle de la racine. Mettons que le gouvernement états-unien, gérant de la racine, la distribue via un site Web avec un mécanisme de vérification. On copie cette clé dans un fichier, mettons `root-trust-anchors` et on indique à Unbound de l'utiliser :

```
trust-anchor-file: "root-trusted-anchors"
```

Le fichier a le format d'un fichier de zone, donc, par exemple (« . » étant la racine) :

```
. IN DNSKEY 257 3 5 AwEAAd+KNrUQaor2JiLB/oodx9HrUjiG...
```

Si on préfère BIND, la syntaxe dans `named.conf` est (notez bien que la clé n'est pas représentée de la même façon) :

```
trusted-keys {
   . 257 3 5 "AwEAAd+KNrUQaor2JiLB/oodx9HrUjiG...";
};
```

Mais ce cas où la racine est signée est trop simple. La racine ne sera pas signée avant la mi-2010 et, de toute façon, cela n'entrainera pas automatiquement la signature de tous les TLD. Il faudra donc probablement gérer plusieurs clés de confiance, ce que permet le format des fichiers cités ci-dessus. Par exemple, avec Unbound :

```
mybank.example. IN DNSKEY 257 3 5 Gfvf56DjkfF...
example.com. IN DNSKEY 257 3 5 oigu7Dd3...
dnssec.fr. IN DNSKEY 257 3 5 ezz3DFn9...
```

[17] http://www.opendnssec.org/

[18] Qu'il stocke dans une base SQLite.

[19] http://www.bortzmeyer.org/dlv-pointpr.html

[20] http://www.unbound.net/

Avec une telle liste, un nom comme web.example.net, qui n'est pas couvert, ne sera pas vérifié[21]. Par contre, web.mybank.example, lui, le sera.

Récupérer les clés et les mettre dans le fichier n'est pas trop compliqué. Bien plus difficile est de suivre ces clés dans le temps. En effet, elles vont être régulièrement remplacées et ce remplacement des clés est l'une des principales causes de problèmes DNSSEC. Comme l'administrateur système typique n'a pas le temps de se tenir au courant de la vie de toutes ces clés, les solutions recommandées sont le RFC 5011, DLV ou la limitation stricte du nombre de clés qu'on gère manuellement.

Le RFC 5011 documente une méthode pour suivre les changements de clé. Si la première clé est bonne et qu'une seconde clé apparaît, signée par la première, BIND accepte désormais la seconde clé. Même chose à la troisième et ainsi de suite. Le RFC 5011dispense donc de gérer les changements de clés de confiance à la main. Avec BIND (il faut au moins la version 9.7), il se configure ainsi :

```
managed-keys {
    "example.com."      initial-key             257 3 5
                                "AwEAAeeGE5unuosN3c8tBcj1/q4TQEwzfNY0GK6kxMVZ ...
```

Ainsi, on met la clé de example.com une seule fois. BIND s'occupera du suivi.

Autre option utile lorsqu'on configure un résolveur, DLV. Avec BIND, cela se fait avec :

```
dnssec-lookaside . trust-anchor dlv.isc.org.;
```

(et il faut aussi mettre la clé de dlv.isc.org dans le bloc trusted-keys).

Et avec Unbound :

```
dlv-anchor-file: "dlv.isc.org.key"
```

Où le fichier contient la clé du registre DLV.

### *5.2.2 La taille compte*

Il reste un sérieux problème. Lors de la création de la norme DNS, la taille maximale des paquets était fixée à 512 octets. Cette limite a été supprimée par le RFC 2671 en 1999. Mais dix ans sont une durée très courte pour le conservatisme de certains et quelques pare-feux mal gérés refusent encore les paquets qu'ils trouvent trop gros. Or, DNSSEC, avec ses signatures, va générer des réponses DNS bien plus grosses que ce à quoi on était habitué. Il y a donc des chances que certains sites se voient coupés du DNS au fur et à mesure du déploiement de DNSSEC. Notamment, la signature de la racine dans la première moitié de 2010, va permettre de détecter les administrateurs système négligents[22].

L'OARC (*DNS Operations, Analysis and Research Center*) a créé un excellent outil pour tester simplement si votre site est dans ce cas. Il consiste simplement en un serveur DNS spécifique, qu'on peut interroger avec n'importe quel client DNS[23]. Voici un exemple avec dig :

```
% dig +short rs.dns-oarc.net txt
...
"192.168.1.1 sent EDNS buffer size 4096"
"192.168.1.1 DNS reply size limit is at least 4023 bytes"
```

Ici, on voit que les réponses DNS de 4023 octets peuvent arriver. Avec des résolveurs DNS mal configurés ou mal connectés (ici, ceux d'Alice via une AliceBox) :

```
% dig +short rs.dns-oarc.net txt
...
"10.228.63.58 lacks EDNS, defaults to 512"
"10.228.63.58 DNS reply size limit is at least 486 bytes"
```

486 octets arrivent à passer, ce qui est insuffisant dans de nombreux cas. Si vous obtenez moins de 2048 octets, danger : DNSEC va vous causer des problèmes.

### *5.2.3 Déboguer*

Une politique de sécurité est un compromis. Trop de sécurité et on ne peut plus rien faire, pas assez et les ennuis nous tombent dessus. Pour se protéger de l'empoisonnement des résolveurs DNS, on pense à déployer DNSSEC. Mais celui-ci entraîne ses propres problèmes. Il existe une longue expérience de déploiement de la cryptographie sur l'Internet et l'une des leçons apprises

[21] En terminologie DNSSEC, il sera « *insecure* », terme assez mal choisi.

[22] http://labs.ripe.net/content/preparing-k-root-signed-root-zone

[23] Son fonctionnement est détaillé en https://www.dns-oarc.net/oarc/services/replysizetest

est que des ennuis se produisent inévitablement. Bogues dans les logiciels et erreurs de procédures de la part des humains font que la cryptographie fonctionne souvent comme une serrure : elle gêne les méchants mais elle peut aussi bloquer les gentils. Normalement, le DNS est très résistant aux erreurs de configuration : il faut vraiment le faire exprès pour rendre une zone complètement inatteignable. Avec DNSSEC, on perd cette robustesse : une erreur et la zone, quoique atteignable, ne sera pas validée et les résolveurs refuseront de transmettre les données. Il est donc crucial d'apprendre à déboguer DNSSEC.

Commençons par un cas banal à l'heure actuelle. On a un résolveur DNS qui valide avec DNSSEC et un matin, une zone DNS n'est plus accessible. Le résolveur, interrogé avec dig dit juste *Server failure* :

```
% dig +dnssec SOA example.net
; <<>> DiG 9.5.0-P2 <<>> +dnssec @dnssec SOA example.net
...
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 31180
...
```

Si on a accès aux journaux de ce résolveur et s'il est configuré en mode suffisamment bavard, on pourra y trouver des détails sur la cause du problème. Mais il faut noter que les résolveurs Unbound et (surtout) BIND ont en commun que les messages ainsi journalisés sont peu compréhensibles, et nécessitent souvent la lecture des RFC pour être décodés.

À l'autre extrémité du spectre de la complexité, une solution entièrement cliquodromesque est le vérificateur du registre de .SE[24], . Si la zone est disponible publiquement, il peut l'analyser en détail et produire des diagnostics compréhensibles par exemple :

```
DNSSEC signature expired: RRSIG(example.net/IN/DNSKEY/8850)
Expired signatures will be ignored by validating resolvers.
```

Cet outil est également distribué, sous une licence libre, pour ceux qui veulent le faire tourner chez eux (les autres outils de vérification du DNS comme Zonecheck[25] n'ont pas encore de tests DNSSEC).

Et entre les deux ? Si on est prêt à utiliser quelques outils Unix mais qu'on n'a pas accès au journal du résolveur, ou bien qu'on ne comprend rien à ses messages ? Mark Andrews, responsable de la version actuelle de BIND, aime dire qu'on peut « déboguer DNSSEC uniquement avec dig et date ». Est-ce vrai ?

Voyons ce qu'on peut obtenir avec dig. Par défaut, la plupart des résolveurs validateurs ne distribuent aucune donnée lorsque la signature est invalide. Cela rend difficile le déboguage aussi faut-il demander gentiment au résolveur d'envoyer quand même les données invalides, avec l'option +cd :

```
% dig +dnssec +cd SOA example.net
...
;; ANSWER SECTION:
example.net.            86400   IN      SOA     ns1.example.net. ...
example.net.            86400   IN      RRSIG   SOA 5 2 86400 \
                20081120064157 20081021064157 8850 example.net. QwzZD+...Y6Q=
```

Le premier enregistrement, le SOA est ce que nous avions demandé, le second, RRSIG est la signature cryptographique. La majorité est du binaire, que je ne chercherai pas à lire mais certaines données sont en clair, notamment l'identité de la clé (ici 8850) et les dates de validité de la signature.

En effet, pour empêcher un attaquant de « rejouer » de vieux enregistrements, les RRSIG ont une date de début et une date de fin de validité[26]. dig affiche cette date dans un format quasiment lisible : YYYYMMDDHHmmSS en UTC. Ici, la date de fin de validité est 20081120064157, donc le 20 novembre 2008 à 06h41 UTC. Comme nous sommes le 26 novembre, pas besoin de chercher plus loin : la signature a expiré[27].

En effet, puisque les signatures ont une durée de vie limitée, il faut re-signer la zone périodiquement. Les futures versions de BIND le feront automatiquement. Mais, en attendant, il faut mettre dans son crontab quelque chose comme :

```
dnssec-signzone -N increment example.net
```

À noter qu'il existe aussi des « boîtes noires » DNSSEC comme celle de Secure64 qui, normalement, vous dispensent de cette tâche.

---

[24] http://dnscheck.iis.se/

[25] http://www.zonecheck.fr/

[26] La période de validité est de un mois, par défaut, si on signe avec l'outil dnssec-signzone de BIND

[27] Pour afficher la date du jour au format identique à celui de dig, sur un Unix qui a GNU date comme Debian on peut faire `date -u +%Y%m%d%H%M%S`.

Mais, pour l'instant, il faut bien dire que la signature expirée est la cause la plus fréquente de problèmes DNSSEC. Tellement qu'elle vaut la peine d'une surveillance spécifique. Par exemple, le script check-sig[28] vérifie un nom de domaine et affiche un message d'erreur si sa signature est expirée ou bien si elle le sera bientôt (sept jours par défaut) :

```
% check-sig example.net
Name example.net has an expired signature (20081120064157)
% check-sig example.org
Name example.org has a signature that will expire in less than 7 days (20081201224442)
```

Il peut donc être mis dans une crontab pour donner l'alarme lorsqu'une signature risque d'expirer.

Et la cause suivante, en nombre d'erreurs ? Probablement les incohérences dans la délégation. DNSSEC repose, comme le DNS, sur un modèle hiérarchique. Une racine délègue à des zones qui délèguent à des sous-zones et ainsi de suite. Chacune de ces délégations, matérialisées par un enregistrement DS, est évidemment signée. La principale différence avec le DNS est que la racine de signature peut être différente de la racine tout court, grâce à DLV. Ainsi, à l'heure actuelle, la plupart des zones signées sont délégués depuis le registre DLV de l'ISC.

Or, des problèmes peuvent survenir lors des délégations puisque elles impliquent deux organisations différentes. Par exemple, un administrateur de zone décide de signer car c'est à la mode puis arrête de le faire mais oublie qu'un enregistrement DLV pointe vers lui (et donc garantit que la zone devrait être signée). Ou bien l'administrateur modifie sa clé et oublie de prévenir la zone au-dessus. De telles contradictions entre la zone parente et la zone fille sont fréquentes dans le DNS d'aujourd'hui mais, avec DNSSEC, elles ne pardonnent pas.

Supposons donc que la zone `example.net` ne fonctionne pas : nous ne récupérons que le *Server Failure*. Regardons sa clé :

```
% dig +dnssec +cd +multi DNSKEY example.net
...
;; ANSWER SECTION:
example.net.            366 IN DNSKEY 257 3 5 (
                            AwEAAc4x/KbNECb+dpDDBSvyxfTlvUxXyC3EAqCnXDp4
                            ...
                            ) ; key id = 17398
```

Sa clé est la 17398. Est-elle bien chez le parent ? On demande à celui-ci :

```
% dig @addr-server-parent +dnssec +cd +multi DS example.net.
...
;; ANSWER SECTION:
                               ; L'identificateur
                               ; vvvv
example.net.            1800 IN DS 6732 5 1 (
                            BBDDDD272C4D81EF941C722CEF79A848B543502D )
```

Oui, il y a bien un enregistrement DS mais pour une autre clé, la 6732. La chaîne de confiance est cassée là. Sans doute un changement de clé effectué sans prévenir le parent. Attention, il est courant d'avoir plusieurs clés et il faut donc les regarder toutes.

Avec DLV, même principe. Ma zone `sources.org` est signée et enregistrée dans DLV à l'ISC, ce qu'on peut voir avec :

```
% dig @ns-ext.isc.org +multi DLV sources.org.dlv.isc.org
...
;; ANSWER SECTION:
sources.org.dlv.isc.org. 3600 IN DLV 22107 5 2 (
                            AF12...3E5 )
sources.org.dlv.isc.org. 3600 IN DLV 14347 3 1 (
                            31F...873 )
sources.org.dlv.isc.org. 3600 IN DLV 14347 3 2 (
                            0DA...A7F )
```

[28] http://www.bortzmeyer.org/files/check-sig.sh

```
sources.org.dlv.isc.org. 3600 IN DLV 22107 5 1 (
                                EA7...775 )
```

Il y a deux clés, les 14347 et 22107, utilisant deux algorithmes de condensation différents, d'où les quatre DLV. Ici, tout va donc bien.

Et si la recherche d'une clé dans la zone ne donnait rien ? C'est que la zone n'est pas signée. Actuellement, c'est l'écrasante majorité. Sauf s'il existe une délégation DNSSEC depuis le parent (la zone serait alors invalide), ces zones ne doivent pas être refusées, sauf extrême paranoïa. Si elles déclenchent un *Server Failure*, c'est qu'il y a une raison non-DNSSEC à cela.

On peut tester ces techniques avec la zone `test.dnssec-tools.org`[29] où se trouvent de nombreux enregistrements DNSSEC cassés volontairement. Par exemple, `futuredate-A.newzsk-ns.test.dnssec-tools.org` a une date des signatures valide seulement dans le futur...

# 6 Qui fait du DNSEC ?

## 6.1 Domaines

La norme actuelle, dite DNSSEC-bis (la précédente n'avait connu aucun déploiement) a été publiée en 2005. Le premier domaine de tête (TLD) signé, .SE (Suède), l'a été en 2007[30]. Le second, .CZ (Tchéquie), en 2008. Le premier gTLD, .GOV, a été signé en 2009, précédant .ORG. Dans tous les cas, la signature n'implique pas que des délégations sécurisées soient possibles. Ainsi, dans .ORG, à l'heure actuelle, les bureaux d'enregistrements ne peuvent pas encore envoyer des enregistrements DS. Les seuls de ces enregistrements qui existent (comme `bondis.org`) sont faits manuellement selon une procédure nommée « F*riends & Family* ».

Certains domaines importants, comme des sous-domaines de `in-addr.arpa` et d'`ip6.arpa`, gérés par le RIPE-NCC, sont également signés.

La racine elle-même, selon le plan annoncé en octobre 2009 à la réunion RIPE de Lisbonne, sera signée à partir de janvier 2010, selon un mécanisme progressif, menant à une signature complète et opérationnelle en juillet. Par contre, aucune date n'a été annoncée pour les délégations sécurisées de TLD (i.e. l'insertion de DS dans la racine).

Du fait du caractère essentiel de la racine, notamment pour la requête initiale du résolveur (le *priming*, où un résolveur qui vient de démarrer essaie d'obtenir la liste à jour des serveurs de noms de la racine), ce déploiement sera à surveiller avec la plus grande attention. Les sites qui continuent à bloquer les réponses DNS de plus de 512 octets (une grave faute, en 2009), pourraient ne plus pouvoir joindre la racine.

L'AFNIC a annoncé la signature de .FR, débutant à l'été 2010, pour un passage en production à l'automne. Des points comme la délégation sécurisée, le rôle des intermédiaires ou bien comme les éventuels coûts n'ont pas encore été tranchés.

Verisign a annoncé la signature de .NET et de .COM en 2011. Cette date tardive s'explique par des blocages politiques ou bureaucratiques, mais aussi par le caractère stratégique de ces gTLD (si .COM est en panne, pour la plupart des utilisateurs, c'est comme s'il n'y avait plus d'Internet) et par leur taille (.COM a plus de soixante-dix millions de domaines, un défi pour les logiciels de signature, et pour les serveurs qui devront charger une telle zone).

## 6.2 Logiciels

Tous les serveurs DNS sérieux[31] aujourd'hui gèrent DNSSEC. C'est le cas, pour les serveurs faisant autorité, de BIND 9 et de NSD, et pour les récurseurs, de BIND 9 et de Unbound. PowerDNS et PowerDNS recursor ont du code DNSSEC dans leurs versions de développement et la version finale devrait sortir en 2010.

Par contre, les logiciels « auxiliaires » comme Zonecheck pour les vérifications de zone, ou comme certains logiciels de gestion et d'édition de zone, n'ont pas encore ces fonctions.

# 7 Conclusion

Pendant longtemps, DNSSEC a semblé être, comme IPv6, une de ces technologies astucieuses, et même brillantes, mais que personne ne déployait car elle ne rapportait rien. Un exposé sur DNSSEC a eu lieu à JRES en 2003 et presque rien n'avait bougé depuis.

Les choses ont toutefois subitement changé avec l'annonce de la faille « Kaminsky » en 2008. Cette faille transformait une vulnérabilité connue, mais très théorique, en une possibilité à court terme. Cette annonce a brusquement débloqué le déploiement de DNSSEC et les annonces se sont suivies de manière régulière depuis. Au début de 2010, il y aura environ dix

[29] `http://www.dnssec-tools.org/testzone/`

[30] Sauf indication contraire, les dates sont toujours pour des mises en production officielle, pas pour des essais ou des démonstrations. .SE avait connu une première signature expérimentale dès 2005.

[31] En excluant les projets personnels.

TLD signés. Mais, en comptant les annonces sérieuses qui ont déjà été faites et en voyant les collègues travailler, on peut dire que ces TLD signés seront cinquante à la fin de 2010, qui sera sans doute l'« année DNSSEC ».

Des grandes questions restent ouvertes : les administrateurs de résolveurs vont-ils activer massivement la validation ? Tout se passera t-il correctement ? Est-ce que la plus grande fragilité du DNS due à DNSEC produira des pannes et un retour de bâton ? De nouveaux progrès dans l'exploitation de la faille Kaminsky imposeront-ils une marche vers DNSSEC encore plus rapide ?

## Bibliographie


[1] P. Vixie, RFC 2671, « *Extension Mechanisms for DNS (EDNS0)* », août 1999

[2] D. Atkins et R. Austein, RFC 3833 « *Threat Analysis of the Domain Name System* », août 2004

[3] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, RFC 4033 « *DNS Security Introduction and Requirements* », mars 2005

[4] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, RFC 4034 « *Resource Records for the DNS Security Extensions* », mars 2005

[5] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, RFC 4035 « *Protocol Modifications for the DNS Security Extensions* », mars 2005

[6] M. StJohns, RFC 5011 « *Automated Updates of DNSSEC Trust Anchors* », septembre 2007

[7] B. Laurie, G. Sisson, R. Arends, D. Blacka, RFC 5155, « *DNSSEC Hashed Authenticated Denial of Existence* », mars 2008

[8] A. Hubert, R. van Mook, RFC 5452, « *Measures for Making DNS More Resilient against Forged Answers* », janvier 2009

[9] Alan Clegg, « *DNSSEC in 6 minutes* ». http://www.isc.org/files/DNSSEC_in_6_minutes.pdf

[10] Olf Kolkman, « *DNSSEC HOWTO* ». http://www.nlnetlabs.nl/dnssec_howto/

[11] SURFnet, « *Hardening the Internet* ». http://www.surfnet.nl/Documents/DNSSSEC-web.pdf