OpenDNSSEC, ou comment faciliter l'utilisation de DNSSEC

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 31 octobre 2009

https://www.bortzmeyer.org/opendnssec-debut.html

Le fait que le DNS soit très vulnérable aux attaques d'empoisonnement (comme l'attaque Kaminsky <https://www.bortzmeyer.org/comment-fonctionne-la-faille-kaminsky.html>) est désormais bien connu. On sait aussi que la seule solution à moyen terme est DNSSEC (mais des solutions à court terme, plus faciles à déployer, existent). Seulement, DNSSEC est un protocole complexe et les erreurs se paient cher, par une impossibilité d'accèder aux informations DNS. Il y a donc en ce moment plusieurs efforts visant à rendre DNSSEC plus digeste, comme par exemple OpenDNSSEC <http://www.opendnssec.org/>.

En quoi DNSSEC, tel que normalisé dans le RFC 4034 [1] et suivants est-il complexe ? Certes, le protocole n'est pas pour ceux qui ont du jus de navet dans les veines (le RFC 5155 étant certainement le pire). Mais, après tout, cela ne concerne que les implémenteurs, les auteurs de logiciels comme BIND ou nsd. Les simples administrateurs système peuvent-ils, eux, ignorer la complexité de DNSSEC ? Pas complètement : en effet, une bonne part de la complexité réside dans les **procédures** qu'il faut suivre pour configurer correctement sa zone et surtout pour qu'elle le reste. En effet, le DNS classique était une statue de chat : une fois une zone configurée (et testée avec des outils comme Zonecheck <http://www.zonecheck.fr>), elle restait configurée. DNSSEC est un chat vivant : il faut aspirer ses poils, le nourrir et changer sa litière. (J'ai oublié qui m'a enseigné cette parabole la première fois. S'il se reconnait...) Quelles sont les tâches récurrentes avec DNSSEC ? Essentiellement la gestion des clés cryptographiques. Il est recommandé de changer ces clés régulièrement (la discussion exacte du **pourquoi** de cette recommandation a déjà fait bouger beaucoup d'électrons, je ne la reprends pas ici). Ce changement est très délicat car, à tout moment, la zone doit être validable. Or, la propagation des informations dans le DNS n'est pas instantanée. Par exemple, en raison des caches chez les résolveurs, notamment des FAI, une signature faite il y a des heures peut être encore disponible... et la clé qui a servi à cette signature doit l'être également. Changer une clé DNSSEC est une opération qui nécessite, soit des procédures manuelles très rigoureuses et parfaitement suivies, soit une automatisation par un logiciel. Et c'est le rôle d'OpenDNSSEC <http://www.opendnssec.org/>.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc4034.txt

Il existe diverses solutions d'automatisation des procédures de changement de clés de DNSSEC. La plupart sont embryonnaires, les problèmes concrets de déploiement de DNSSEC sont assez récents (le RFC 6781 en donne une idée). Il y a des solutions sous forme d'un boîtier fermé (comme Secure64) ou bien en logiciel libre comme OpenDNSSEC. Ce dernier est encore en version béta et je préviens donc tout de suite il faut être prêt à bricoler un peu.

J'ai rédigé cet article en travaillant sur une machine Debian. OpenDNSSEC nécessite plusieurs programmes tiers et voici une liste non-exhaustive de ce qu'il faudra installer :

```
% sudo aptitude install libxml2-dev rubygems libopenssl-ruby \
          libsqlite3-dev libldns-dev libdns-ruby
```

Ensuite, il faut choisir le mécanisme de stockage des clés. La méthode la plus sûre est d'utiliser un HSM ("*Hardware Security Module*"), un boîtier qui génère les clés, les stocke, et fait les signatures cryptographiques. Certains HSM (à partir de la certification FIPS 140-3) sont en outre résistants aux attaques physiques : si vous essayez de les ouvrir, la clé privée est détruite. Vos clés sont ainsi parfaitement en sécurité.

Il existe une API standard pour accéder à ces HSM, PKCS #11 et la plupart des logiciels DNSSEC (comme, par exemple, le signeur de BIND) la gèrent. En pratique, toutefois, il semble que cela ne soit pas toujours aussi simple et on peut trouver facilement sur le Web beaucoup de récits désabusés...

En outre, ces HSM sont des engins relativements chers (apparemment à partir de 1 000 € en France) et difficiles à obtenir. Faites l'expérience avec des vendeurs nationaux de HSM comme Thales ou Bull en France : les commerciaux ne vous rappellent pas. Pour commencer, on va donc utiliser un stockage logiciel des clés, dans le système appelé SoftHSM <http://trac.opendnssec.org/wiki/SoftHSM> (oui, c'est un oxymore), dû aux même auteurs que OpenDNSSEC. SoftHSM permet de stocker les clés en local, dans une base SQLite (qu'il faudra évidemment protéger avec soin, le matériel ne le faisant pas pour vous).

Bref, on installe SoftHSM, on n'oublie pas, comme la documentation l'indique, le `ldconfig` pour que les nouvelles bibliothèques soient trouvées, on initialise notre « HSM » :

```
% softhsm --init-token --slot 0 --label "OpenDNSSEC"
```

On devra alors, comme avec un vrai HSM, indiquer un « PIN », un mot de passe.

Le HSM étant prêt, on peut compiler et installer OpenDNSSEC. On édite son `/etc/opendnssec/conf.xml` pour indiquer le PIN :

```
        <Repository name="softHSM">
               <Module>/usr/local/lib/libsofthsm.so</Module>
               <TokenLabel>OpenDNSSEC</TokenLabel>
               <PIN>cestressecret</PIN>
        </Repository>
```

OpenDNSSEC est alors prêt à utiliser ce « HSM ». Il connait le nom de la bibliothèque à charger pour lui parler (`/usr/local/lib/libsofthsm.so`) et le PIN. Testons qu'il apparait bien dans la liste des HSM :

```
% sudo hsmutil list
hsm_session_init(): Incorrect PIN for repository softHSM
```

OK, faute de frappe en tapant le PIN, on recommence :

```
% sudo hsmutil list
Listing keys in all repositories.
0 key found.

Repository            ID                                Type
----------            --                                ----
```

C'est parfait. (Dans les versions d'OpenDNSSEC ultérieures, toutes les commandes sont préfixées de `ods-` donc on aurait écrit `sudo ods-hsmutil list`.) Il n'y a pas de clé mais, à ce stade, c'est normal (OpenDNSSEC les créera tout seul).

Maintenant, on configure OpenDNSSEC : quelles zones doit-il gérer, avec quels paramètres, etc. Les fichiers de configuration livrés contiennent déjà des paramètres par défaut (par exemple, clé RSA de 2048 bits pour la KSK, la "*Key Signing Key*", périodes avant le renouvellement des clés, en suivant la syntaxe décrite en `<http://trac.opendnssec.org/wiki/Signer/Using/Configuration>`, etc). Je ne touche pas tout de suite à `/etc/opendnssec/kasp.xml`, qui contient les politiques de définition de ces paramètres et j'édite `/etc/opendnssec/zonelist.xml` qui contient la liste de mes zones :

```
   <Zone name="bortzmeyer.fr">
        <Policy>default</Policy>
        ...
        <Adapters>
               <Input>
                    <File>/var/opendnssec/unsigned/bortzmeyer.fr</File>
               </Input>
               <Output>
                    <File>/var/opendnssec/signed/bortzmeyer.fr</File>
               </Output>
...
```

Cette zone sera signée avec les paramètres par défaut (définis dans `kasp.xml`) et je dois mettre le fichier de zone non signé dans `/var/opendnssec/unsigned/bortzmeyer.fr`. (Attention pour ceux qui ont déjà utilisé le signeur `dnssec-signzone` de BIND : ici, on ne doit **pas** inclure les DNSKEY, OpenDNSSEC fait cela tout seul.)

Je compile les politiques (`sudo ksmutil setup` la première fois, `sudo ksmutil update all` ensuite ; comme indiqué plus haut, c'est `ods-ksmutil` dans les OpenDNSSEC récents). Je démarre alors les démons OpenDNSSEC :

```
% sudo ./ods-signerd
% sudo ./ods-enforcerd
```

Le premier est le signeur. Écrit avec la bibliothèque ldns <http://www.nlnetlabs.nl/projects/ldns/>, il réalise les opérations de signature (avec un vrai HSM, elles seraient largement déléguées au matériel). Le second démon, le policeur, fait respecter les politiques de remplacement des clés qu'on a définies.

Normalement, le policeur doit alors noter qu'il n'a pas de clés, et les générer. Vérifions :

```
% sudo hsmutil  list
Listing keys in all repositories.
4 keys found.

Repository            ID                                Type
----------            --                                ----
softHSM               cc59d2b16e421c57eec35ed4ceae0099  RSA/1024
softHSM               e17b1d0465e6976536119c872a353911  RSA/1024
softHSM               fa8cdfc8da319311283b66fe55839212  RSA/2048
softHSM               60b57dff6604cc35ec6fdd8aef7710a2  RSA/2048
```

C'est parfait, on a deux ZSK (1024 bits par défaut) et deux KSK (2048 bits par défaut), une active et une prête à assurer le remplacement futur. On la publie en avance, pour être sûr qu'elle soit disponible dans les caches DNS.

De la même façon, la zone a dû être signée et le résultat mis en `/var/opendnssec/signed/bortzmeyer.fr`. Sinon, il faut regarder le journal pour savoir ce qui s'est passé. On peut aussi forcer OpenDNSSEC à signer tout de suite avec `ods-signer sign bortzmeyer.fr`. Par défaut, OpenDNSSEC ne fait pas le travail si la zone n'a pas changé pour on peut le convaincre de signer quand même avec `ods-signer clear bortzmeyer.fr` qui précède le `ods-signer sign bortzmeyer.fr`.

OpenDNSSEC permet de gérer plusieurs zones DNS donc si je veux voir uniquement les clés d'une zone, je me sers de `--zone` :

```
% sudo ksmutil key list  --zone bortzmeyer.fr
SQLite database set to: /var/opendnssec/kasp.db
Keys:
Zone:                           Keytype:      State:    Date of next transition:
bortzmeyer.fr                   KSK           active    2010-10-27 13:14:39
bortzmeyer.fr                   KSK           ready     next rollover
bortzmeyer.fr                   ZSK           active    2009-11-26 13:14:39
bortzmeyer.fr                   ZSK           ready     next rollover
```

Je vois ainsi les clés, et le moment où elles seront remplacées (par la clé qui est dans l'état `ready`). `--verbose` affiche également le *"key tag"* (annexe B du RFC 4034), également mis par le signeur dans le fichier de zones et qu'affiche aussi dig avec l'option `+multi`, ce qui est pratique pour déboguer les zones une fois chargées :

```
% sudo ksmutil key list  --zone bortzmeyer.fr --verbose
SQLite database set to: /var/opendnssec/kasp.db
Keys:
Zone:                           Keytype:      State:   ... Keytag:
bortzmeyer.fr                   KSK           active   ... 24045
bortzmeyer.fr                   KSK           ready    ... 36168
...
% dig @192.0.2.1 +dnssec +multi DNSKEY bortzmeyer.fr
bortzmeyer.fr.          3600 IN DNSKEY 257 3 7 (
                                ...
                                ) ; key id = 24045
```

24045 étant le "*key tag*" de la KSK active.

OpenDNSSEC ne permet pas de mettre à jour automatiquement la clé dans la zone parente (en partie parce qu'il existe plusieurs protocoles différents pour cela, comme celui du RFC 4310). Mais il aide, en indiquant ce qu'il faut transmettre au parent, un enregistrement DS :

```
% sudo ksmutil key export --ds
SQLite database set to: /var/opendnssec/kasp.db

;active KSK DS record (SHA1):
bortzmeyer.fr.  3600    IN      DS      24045 7 1 c57467e055adde7fcad11...

;active KSK DS record (SHA256):
bortzmeyer.fr.  3600    IN      DS      24045 7 2 2d47e39d7f04237292760...
```

Seule la KSK actuellement active, la 24045, est ainsi « exportée » (avec deux algorithmes de hachage possibles).

OpenDNSSEC n'a pas de problèmes avec les petites zones. Si on essaie de signer des zones de plus d'un million d'enregistrements, il vaut mieux supprimé l'auditeur, encore trop bogué. L'auditeur a pour but de vérifier l'intégrité du fichier signé et sa conformité aux politiques décrites dans `kasp.xml`. S'il échoue, la zone signée n'est donc pas publiée. Comme il échoue souvent, je l'ai pour l'instant débrayé : suppression de `<Auditor>` dans `conf.xml`, de `<Audit>` dans `kasp.xml` et `ods-ksmutil update` pour prévenir les démons qui sont en cours d'exécution. Les signatures se dérouleront alors normalement.

Prochaines étapes, chers lecteurs :
— Voir si on peut utiliser OpenDNSSEC avec les mises à jour DNS dynamiques (il semble que non),
— Tester avec un vrai HSM,
— Faire marcher l'auditeur.