Installer et utiliser OpenDNSSEC.

Laurent Archambault <archi.laurent@gmail.com>

Installer et utiliser OpenDNSSEC.

par Laurent Archambault

Table des matières

Contexte

Le but de ce document est de montrer comment activer un service DNS sous Bind en utilisant les paramètres DNSSEC. DNSSEC existe depuis bien longtemps (mars 1999 et le RFC 2535), son déploiement est tout de même difficile à mettre en oeuvre. Et puis tout allait bien, pourquoi se compliquer la vie, jusqu'à l'attaque DNS récente en 2008 (DNS cache poisoning [http://cert.lexsi.com/weblog/index.php/2008/07/10/249-vulnerabilite-permettant-des-attaques-de-dns-cache-poisoning]).

De nos jours, de gros efforts sont faits pour automatiser son utilisation et son déploiement. Il existe de nombreuses solutions. Personnellement j'ai préféré ce projet à d'autres, en parlant de OpenDNSSEC. Son déploiement est relativement simple, mais nécessite une documentation, indispensable mes yeux. Sur le site de OpenDNSSEC, est présente une documentation claire et précise en anglais.

Le site www.dnssec.net [http://www.dnssec.net/software] recense beaucoup de solutions et d'informations à propos de DNSSEC. Ce site est pour moi, une sorte de « Wikipédia (GB) » pour DNSSEC, je ne peux que conseiller cette visite.

Ce document est destiné à des administrateurs qui possèdent déjà une bonne maîtrise de la gestion de zones avec le logiciel Bind. Il est aussi nécessaire de connaître au moins les grands principes de DNSSEC, pour aborder ce document.

Avertissement

Le 11 février 2010, la version d'OpenDNSSEC vient de passer en version « stable » (opendnssec-1.0.0). Je ne peux hélas refaire toute ma documentation. Néanmoins j'ai adapté le téléchargement des diverses librairies à cette « contrainte ».

Chapitre 1. Installer les composants.

Les composants via une distribution Gnu/Linux.

Les composants qui suivent s'installent très simplement; ils sont indispensables. Ils doivent être tous disponibles sur la plupart des distributions Gnu/linux sans problème. *OpenSSL* et d'autres programmes peuvent évidemment être compilés à part.

Le 25 janvier 2010, l'équipe « OpenDNSSEC » à procéder à deux mises à jour importantes. Les versions *opendnssec 1.0.RC3* et *softhsm 1.1.3* sont disponibles. J'ai mis à jour ma documentation, en particulier pour les paramètres d'installation, sans modifier les exemples. Donc ne soyez pas surpris de rencontrer des anciennes versions, les modifications ne portent pas sur les commandes, mais le sur fonctionnement intrinsèque des logiciels.

```
apt-get install g++ libopenssl-ruby1.9 subversion autoconf automake libtool \
libxml2 libxml2-dev openssl libssl-dev sun-java6-jre sun-java6-plugin \
sun-java6-fonts sqlite3 libsqlite3-dev python-4suite-xml ruby rubygems1.9 \
libruby-extras
```

Cette commande relativement longue peut bloquer, il vous reste sinon le traitement individuel au cas par cas pour débloquer la situation.

Les autres composants.

Placez-vous dans un répertoire dédié pour l'installation de *OpenDNSSEC*, puis faites les étapes suivantes dans l'ordre.

dnsruby

Il est nécessaire d'aller sur le site : rubyforge.org/projects/dnsruby/ [http://rubyforge.org/projects/dnsruby/], puis de télécharger la dernière version. A ce jour, la version disponible est la version 1.43. Attention, OpenDNSSEC n'est pas encore stable et parfois nécessite une version ultérieure. N'hésitez pas à visiter le site opendnssec.org [http://www.opendnssec.org/], pour récupérer la dernière version adéquate. Une fois téléchargé (attention ce lien change régulièrement (version): dnsruby-1.42.gem ou ultérieure [http://rubyforge.org/projects/dnsruby/]), lancez cette commande, et c'est tout :

```
wget http://rubyforge.org/frs/download.php/68933/dnsruby-1.43.gem ;\
                sudo gem install dnsruby-1.43.gem
```

Cette commande installe ce qu'il faut automatiquement, sauf parfois... Sur ma deuxième machine, cette même librairie refusait de s'installer...après maints essais et recherches, une mise à jour de *gem* a permis de résoudre ce problème. La mise à jour est disponible également sur le même site, voici la commande :

gem install rubygems-update-1.3.5.gem.

ldns

Cette librairie est utilisée par le logiciel Unbound [28], qui lui aussi est dédié à DNSSEC. Téléchargez la dernière version stable. Cette librairie est disponible sur ce site :

http://www.nlnetlabs.nl/downloads/ldns/ldns-1.6.4.tar.gz

```
tar -xvf ldns-1.6.4.tar.gz ; cd ldns-1.6.4 ; ./configure && make && make install
```

Voilà, c'est tout à ce niveau. A signaler tout de même que cette librairie inclut des logiciels dédiés au DNS/DNSSEC. Ils peuvent être compilés et installés depuis le répertoire Drill [28].

Botan

Cette librairie est dédiée aux chiffrements. Elle est utilisée en particulier par OpenDNSSEC pour l'un de ses composants. Pour la récupération de cette librairie, il suffit d'aller sur ce site, et comme pour les autres de prendre la dernière version. files.randombit.net/botan [http://files.randombit.net/botan/]. Cette librairie évolue souvent

lors de la rédaction de ce document. La version stable était la 1.9.3. Attention, le fichier téléchargé se termine parfois en « .tbz », et non « .tgz ».

```
wget http://files.randombit.net/botan/v1.9/Botan-1.9.3.tbz ; \
tar -xvpf Botan-1.9.3.tbz ; cd Botan-1.9.3/
```

Il faut compiler tout via les commandes suivantes :

```
./configure.py && make && make install
```

Les composants « OpenDNSSEC ».

Ces composants sont dans ma situation au nombre de deux. L'un des logiciels remplace ou simule un HSM, sa mission est de générer et stocker les clés. Ce logiciel s'appelle « Softhsm ».

OpenDNSSEC à besoin d'une base de données pour le stockage des clés et pour d'autres paramètres. Vous avez le choix entre *`Sqlite3`* et *`Mysql`*. Le support complet de *`Mysql`* est arrivé dans ce projet qu'après. J'ai commencé par obligation avec « Sqlite3 ». N'ayant rencontré aucun problème avec ce logiciel, je n'ai pas changé. Et tout fonctionne très bien, néanmoins mes zones sont limitées et petites.

OpenDNSSEC.

Il faut télécharger la dernière version sur le site de OpenDNSSEC.org [http://www.opendnssec.org]. A ce jour c'est la version *`1 RC3`*. Cette version est stable malgré son « Release Candidate 3 »...n'ayez pas peur avec. Voici le lien, et les diverses commandes nécessaires pour la compilation :

```
wget http://www.opendnssec.org/files/source/opendnssec-1.0.0rc3.tar.gz
~# tar -xvzpf opendnssec-1.0.0rc3.tar.gz ; \
~# cd opendnssec-1.0.0rc3 ; \
./configure --prefix=/usr/local --host=i686-pc-linux-gnu --build=i686-pc-linux-gnu \
--localstatedir=/var --with-botan=/usr/local/bin --sysconfdir=/etc \
--with-openssl=/usr/local/ssl && \
make && make install
```

Bien évidemment il ne faut pas d'erreur lors de cette compilation. L'activation de l'option « *`--with-ldns=/usr/local/bin`* » est facultative, de plus elle peut provoquer une erreur due à une mauvaise recherche sur son numéro de version.

Il est nécessaire de mettre un peu la main à la pâte (éventuellement) pour vérifier que les librairies et les binaires seront bien reconnus là où ils sont. Il faut vérifier dans le fichier */etc/ld.so.conf*, que soit inscrit le répertoire */usr/local/lib*. Pour une Debian|Ubuntu, ceci se résume à :

```
cat /etc/ld.so.conf.d/libc.conf
# libc default configuration
/usr/local/lib
```

Cette mise à jour doit s'effectuer par la commande suivante : « **ldconfig** ». Cette même commande est à utiliser à mon avis par sécurité à la fin de l'installation de chaque composant.

Nous avons fait le nécessaire pour les librairies, les fichiers binaires doivent eux aussi être localisés rapidement. Il suffit de vérifier la prise en compte des répertoires suivants : */usr/local/bin* et */usr/local/sbin* dans votre fichier d'environnement. Pour une Ubuntu, voici comment vérifier (Gentoo = /etc/profile.env):

```
cat /etc/environment
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games"
```

Softhsm (HSM)

La dernière version de ce logiciel est la « 1.1.3 », il est stable et ne pose aucun problème. Il faut télécharger sur le site OpenDNSSEC.org [http://www.opendnssec.org], le fichier suivant ou la version ultérieure : http://www.opendnssec.org/files/source/softhsm-1.1.3.tar.gz.

Voici la procédure pour sa compilation :

```
wget http://www.opendnssec.org/files/source/softhsm-1.1.3.tar.gz ; \
tar -xzvpf softhsm-1.1.3.tar.gz
```

```
./configure \
--prefix=/usr/local \
--host=i686-pc-linux-gnu \
--build=i686-pc-linux-gnu \
--localstatedir=/var \
--sysconfdir=/etc && make && make install
```

Le fichier de configuration */etc/softhsm.conf* doit pouvoir être trouvé au travers des variables d'environnement. Ce n'était pas mon cas lors de la rédaction de ce chapitre, et le système fonctionnait bien. Néanmoins, il est toujours demandé de le faire, chose faite pour moi désormais.

```
echo "export SOFTHSM_CONF=\"/etc/softhsm.conf\"" >> /etc/profile ; source /etc/profile
```

C'est la fin de la partie installation, et pour finir, faites une fois de plus un : **ldconfig**, il sera nécessaire pour la suite.

Les fichiers installés.

Voici les composants qui ont été installés dans le disque dur :

Les librairies installées :

```
/usr/local/lib
-rw-r--r-- 1 root root   107326 2009-12-05 06:58 libhsm.a
-rwxr-xr-x 1 root root      965 2009-12-05 06:58 libhsm.la
lrwxrwxrwx 1 root root       15 2009-12-05 06:58 libhsm.so.0 -> libhsm.so.0.1.1
-rwxr-xr-x 1 root root    98334 2009-12-05 06:58 libhsm.so.0.1.1
-rw-r--r-- 1 root root   770546 2009-11-05 06:59 libsofthsm.a
-rwxr-xr-x 1 root root     1024 2009-11-05 06:59 libsofthsm.la
lrwxrwxrwx 1 root root       19 2009-11-05 06:59 libsofthsm.so.1 -> libsofthsm.so.1.0.1
-rwxr-xr-x 1 root root   454557 2009-11-05 06:59 libsofthsm.so.1.0.1
```

Il est possible que se côtoient des anciennes et des nouvelles versions de librairies, dans cette liste.

Les binaires installés :

```
/usr/local/bin/
lrwxrwxrwx 1 root root     12 2009-12-05 06:58 hsmspeed -> ods-hsmspeed
lrwxrwxrwx 1 root root     11 2009-12-05 06:58 hsmutil -> ods-hsmutil
lrwxrwxrwx 1 root root     11 2009-12-05 06:58 ksmutil -> ods-ksmutil
-rwxr-xr-x 1 root root   3058 2009-12-05 06:58 ods-auditor
-rwxr-xr-x 1 root root   2029 2009-12-05 06:58 ods-control
-rwxr-xr-x 1 root root  27114 2009-12-05 06:58 ods-hsmspeed
-rwxr-xr-x 1 root root  39924 2009-12-05 06:58 ods-hsmutil
-rwxr-xr-x 1 root root   2121 2009-12-05 06:58 ods-kaspcheck
-rwxr-xr-x 1 root root 369556 2009-12-05 06:58 ods-ksmutil
-rwxr-xr-x 1 root root   4984 2009-12-05 06:58 ods-signer
-rwxr-xr-x 1 root root 115213 2009-11-05 06:59 softhsm
-rwxr-xr-x 1 root root 152199 2009-11-05 06:59 softhsm-keyconv

/usr/local/sbin/
-rwxr-xr-x 1 root root 305632 2009-12-05 06:58 ods-enforcerd
-rwxr-xr-x 1 root root   1486 2009-12-05 06:58 ods-signerd

/usr/local/libexec/
drwxr-xr-x 2 root root 4096 2009-12-05 06:58 opendnssec
```

Il est important de présenter les composants principaux qui font d'OpenDNSSEC un excellent projet.

- **ods-auditor**

  Ce programme est chargé de vérifier la conformité des zones avant une intervention. Ce programme peut être lancé manuellement. OpenDNSSEC vérifie automatiquement les zones avant tout traitement. Le résultat est visible sur l'écran, et est inscrit dans les historiques via Syslog.

  ```
  Usage: ods-auditor [options]
  Specific options:
      -c, --conf [PATH_TO_CONF_FILE]   Path to OpenDNSSEC configuration file
  ```

```
                                        (defaults to /etc/opendnssec/conf.xml)
    -k, --kasp [PATH_TO_KASP_FILE]   Path to KASP policy file
                                        (defaults to the path given in the configuratio
    -z, --zone [ZONE_NAME]           Single zone to audit
                                        (defaults to audit all zones)
    -s [PATH_TO_SIGNED_FILE]         If a single zone is specified, then this option ma
        --signed                        the specified signed file with another. This is
                                        the signer.
                                        (defaults to the path given in the zone list)
    -v, --version                    Display version information
Common options:
    -h, -?, --help                   Show this message
```

- **ods-control**

  Ce programme est chargé de lancer ou d'arrêter les programmes résidents (daemons) nécessaires au maintien des zones signées. Régulièrement des contrôles sont effectués. Les résultats sont inscrits dans le fichier des historiques génériques via Syslog.

```
# ods-control -h
usage: ods-control ksm|hsm|signer|start|stop ...
```

- **ods-hsmspeed**

  Sa fonction est de tester la rapidité de votre système lors de la création de clés. Je n'ai personnellement pas testé avec grand intérêt, et pourtant il est capital pour les gestionnaires de zones très importantes. Le délai de traitement pour le chiffrement doit impérativement être pris en compte en particulier pour le remplacement des clés, et leurs diffusions.

```
# ods-hsmspeed -h
ods-hsmspeed: invalid option -- 'h'
usage: ods-hsmspeed [-c config] -r repository [-i iterations] [-s keysize] [-t threads
```

- **ods-hsmutil**

  Ce programme est activé uniquement par OpenDNSSEC, il peut lister les clés comme en supprimer ou en créer...

```
# ods-hsmutil -h
usage: ods-hsmutil [-c config] [-v] command [options]
  list [repository]
  generate <repository> rsa <keysize>
  remove <id>
  purge <repository>
  dnskey <id> <name>
  test <repository>
```

- **ods-kaspcheck**

  Apparemment, ce programme vérifie la conformité du fichier `kasp.xml`.

- **ods-ksmutil**

  Ce programme a de multiples fonctions sur la gestion des zones. Ce programme peut ajouter, mettre à jour ou supprimer une zone; il peut aussi lister les clés. C'est l'un des composants les plus importants et les plus utilisés. Il fonctionne par passage de variables au lancement, mais aussi directement en ligne de commande.

```
# ods-ksmutil -h
usage: ods-ksmutil [-f config] command [options]

  setup
 Import config into a database (deletes current contents)
  update kasp
  update zonelist
  update conf
  update all
 Update database from config
  zone add
 --zone <zone>                          aka -z
 [--policy <policy>]                    aka -p
```

```
 [--signerconf <signerconf.xml>]          aka -s
 [--input <input>]                        aka -i
 [--output <output>]                      aka -o
  zone delete
 --zone <zone> | --all                    aka -z / -a
  zone list
  repository list
  policy export
 --policy [policy_name] | --all           aka -p / -a
  policy list
  key list
 [--verbose]
 --zone <zone> | --all                    aka -z / -a
  key export
 --zone <zone> | --all                    aka -z / -a
 [--keystate <state>]                     aka -e
 [--keytype <type>]                       aka -t
 [--ds]                                   aka -d
  key import
 --cka_id <CKA_ID>                        aka -k
 --repository <repository>                aka -r
 --zone <zone>                            aka -z
 --bits <size>                            aka -b
 --algorithm <algorithm>                  aka -g
 --keystate <state>                       aka -e
 --keytype <type>                         aka -t
 --time <time>                            aka -w
 [--retire <retire>]                      aka -y
  key rollover
 --zone zone [--keytype <type>]           aka -z
  key rollover
 --policy policy [--keytype <type>]       aka -p
  key purge
 --zone <zone>                            aka -z
  key purge
 --policy <policy>                        aka -p
  key generate
 --policy <policy>
 --interval <interval>
  key ksk-roll
 --zone <zone>                            aka -z
 --keytag <keytag> | --cka_id <CKA_ID>    aka -x / -k
  backup done
 --repository <repository>                aka -r
  backup list
 --repository <repository>                aka -r
  rollover list
 [--zone <zone>]
  database backup
 [--output <output>]                      aka -o
key states: GENERATED|PUBLISHED|READY|ACTIVE|RETIRED|REVOKED|DEAD
key types:  KSK|ZSK

 Allowed date/time strings are of the form:
 YYYYMMDD[HH[MM[SS]]]               (all numeric)

 or  D-MMM-YYYY[:| ]HH[:MM[:SS]]     (alphabetic  month)
 or  DD-MMM-YYYY[:| ]HH[:MM[:SS]]    (alphabetic  month)
 or  YYYY-MMM-DD[:| ]HH[:MM[:SS]]    (alphabetic month)

 D-MM-YYYY[:| ]HH[:MM[:SS]]          (numeric month)
 DD-MM-YYYY[:| ]HH[:MM[:SS]]         (numeric month)
 or  YYYY-MM-DD[:| ]HH[:MM[:SS]]     (numeric month)

 ... and the distinction between them is given by the location of the
 hyphens
```

- **ods-signer**

Ce programme est chargé de signer les zones. A ne pas confondre avec le même style de programme qui travaille en mode résident (ods-signerd) .

```
# ods-signer -h
Usage: ods-signer [options] command

The command 'help' shows a complete list of commands

Options:
  -h, --help            show this help message and exit
  -a, --all             when using the command sign, sign all zones
  -f FILE, --file=FILE  connect to domain socket <file>
```

- **softhsm**

Ce programme est chargé de faire le lien entre OpenDNSSEC et un HSM. Dans ma situation, cette tâche est réalisée par lui-même. Voir le chapitre « Création des bases de données[18] » pour la réalisation. Cette action nécessite une base de données spécifique, elle est protégée par un code PIN. Ce logiciel utilise le fichier de configuration conf.xml pour ses paramètres.

```
# softhsm -h
Support tool for libsofthsm
Usage: softhsm [OPTIONS]
Options:
  --show-slots      Display all the available slots.
  --init-token      Initialize the token at a given slot.
                    Use with --slot, --label, --so-pin, and --pin.
                    WARNING: Any content in token token will be erased.
  --import <path>   Import a key pair from the given path.
                    The file must be in PKCS#8-format.
                    Use with --file-pin, --slot, --label, --id and --pin.
  --export <path>   Export a key pair to the given path.
                    The file will be written in PKCS#8-format.
                    Use with --file-pin (will encrypt file), --slot, --id
                    and --pin.
  --file-pin <PIN>  Supply a PIN if the file is encrypted.
  --force           Override some warnings.
  -h                Shows this help screen.
  --help            Shows this help screen.
  --id <hex>        Defines the ID of the object. Hexadecimal characters.
                    Use with --force if multiple key pairs may share
                    the same ID.
  --label <text>    Defines the label of the object or the token.
  --pin <PIN>       The PIN for the normal user.
  --slot <number>   The slot where the token is located.
  --so-pin <PIN>    The PIN for the Security Officer (SO).

You also need to have a configuration file to specify path to the token databases (defa
The path to the configuration file can be changed by the SOFTHSM_CONF environment varia
    export SOFTHSM_CONF=/home/user/config.file

An example of a configuration file:
    0:/home/user/my.db
    # Comments can be added
    # Format:
    # <slot number>:<path>
    4:/home/user/token.database
```

- **softhsm-keyconv**

Ce logiciel porte très bien son nom; il est dédié à la migration d'une zone gérée depuis Bind (en particulier) vers OpenDNSSEC. Il peut aussi faire l'inverse, exporter des clés vers un autre serveur DNS.

```
# softhsm-keyconv -h
Converting between BIND .private-key format and PKCS#8 key file format.
Usage: softhsm-keyconv [OPTIONS]
Options:
```

```
  --topkcs8           Convert from BIND .private-key format to PKCS#8.
                          Use with --in, --out, and --pin.
  --tobind            Convert from PKCS#8 to BIND .private-key format.
                          Use with --in, --pin, --name, --ttl, --ksk, and --algorithm.
  --algorithm <alg>   Specifies which DNSSEC algorithm to use in file.
                           RSAMD5
                           DSA
                           RSASHA1
                           RSASHA1-NSEC3-SHA1
                           DSA-NSEC3-SHA1
                           RSASHA256
                           RSASHA512
  -h                  Shows this help screen.
  --help              Shows this help screen.
  --in <path>         The path to the input file.
  --ksk               Set the flag to 257. Key Signing Key. Optional.
  --name <name>       The owner name. Do not forget the trailing dot, e.g. "example.com
  --out <path>        The path to the output file.
  --pin <PIN>         To encrypt/decrypt PKCS#8 file. Optional.
  --ttl <ttl>         The TTL to use for the DNSKEY RR. Optional.

  The following files will be created:
    K<name>+<alg id>+<key tag>.key Public key in RR format
    K<name>+<alg id>+<key tag>.private Private key in key format
  E.g.
    Kexample.com.+007+05474.private
```

- **ods-enforcerd**

  Ce programme fonctionne en mode résident. Il est chargé régulièrement de maintenir les zones signées à jour. Il se sert des instructions du fichier conf.xml pour réaliser en tache de fond ses opérations.

```
# ods-enforcerd -h
Usage: ods-enforcerd [OPTION]...
OpenDNSSEC Enforcer version 1.0.0rc1

Supported options:
  -c <file>   Use alternate conf.xml.
  -d          Debug.
  -1          Run once, then exit.
  -P pidfile  Specify the PID file to write.
  -v          Print version.
  -[?|h]      This help.
```

- **ods-signerd**

  Également, il porte bien son nom, sa mission est de signer une ou plusieurs zones. Ce programme ne fonctionne qu'en mode résident, sous les ordres d'OpenDNSSEC.

```
# ods-signer -h
Usage: ods-signer [options] command
The command 'help' shows a complete list of commands

Options:
  -h, --help            show this help message and exit
  -a, --all             when using the command sign, sign all zones
  -f FILE, --file=FILE  connect to domain socket <file>
```

Autres composants installés :

La liste des fichiers installés n'apporte que peu d'intérêt. Il s'agit pour la plupart de fichiers au format XML. Il faut noter tout de même la présence du fichier « `database_create.sqlite3` », celui ci peut servir pour initialiser une base de données Sqlite3 (OpenDNSSEC), ou pour voir la structure des tables.

```
/usr/local/share/opendnssec/
-rw-r--r-- 1 root root  4637 2009-12-05 06:58 conf.rnc
-rw-r--r-- 1 root root  8674 2009-12-05 06:58 conf.rng
-rw-r--r-- 1 root root 13578 2009-12-05 06:58 database_create.sqlite3
-rw-r--r-- 1 root root  9644 2009-12-05 06:58 kasp2html.xsl
```

```
-rw-r--r-- 1 root root  5599 2009-12-05 06:58 kasp.rnc
-rw-r--r-- 1 root root  9910 2009-12-05 06:58 kasp.rng
-rw-r--r-- 1 root root  3498 2009-12-05 06:58 signconf.rnc
-rw-r--r-- 1 root root  6019 2009-12-05 06:58 signconf.rng
-rw-r--r-- 1 root root  2349 2009-12-05 06:58 zonefetch.rnc
-rw-r--r-- 1 root root  3665 2009-12-05 06:58 zonefetch.rng
-rw-r--r-- 1 root root  2052 2009-12-05 06:58 zonelist.rnc
-rw-r--r-- 1 root root  2780 2009-12-05 06:58 zonelist.rng
```

Les fichiers installés pour la configuration :

Les répertoires suivants serviront pour la gestion des zones DNS. A ce sujet, et à part le répertoire *`unsigned`*, où j'ai placé mes fichiers de zones « vierges », je ne suis à aucun moment intervenu sur les fichiers présents dans ses différents répertoires. Il est facile de deviner la vocation de ses sous répertoires.

```
/var/opendnssec/tmp
/var/opendnssec/signconf
/var/opendnssec/unsigned
/var/opendnssec/signed
```

Ce répertoire est chargé d'enregistrer les fichiers nécessaires des programmes résidents (socket et PID).

```
/var/run/opendnssec
 rw-r--r-- 1 root root 5 2010-01-18 07:22 enforcerd.pid
srwxr-xr-x 1 root root 0 2010-01-18 07:22 engine.sock
```

Pour mémoire, la plupart des fichiers de configuration sont stockés dans « */etc/opendnssec* ». C'est le seul fichier à être séparé des autres. Ce fichier doit recenser au moins une « connexion (slot) » vers une base de données, ou plusieurs (sqlite3).

```
/etc/
-rw-r--r-- 1 root root 84 2009-11-05 12:13 /etc/softhsm.conf
```

Ce répertoire rassemble les fichiers de configuration qui servira plus tard à OpenDNSSEC. Les fichiers avec l'extension « .sample » peuvent être effacés sans problème.

```
/etc/opendnssec/
-rw-r--r-- 1 root root 1746 2009-11-08 08:06 conf.xml
-rw-r--r-- 1 root root 1699 2009-12-05 06:58 conf.xml.sample
-rw-r--r-- 1 root root 1665 2009-11-21 08:02 kasp.xml
-rw-r--r-- 1 root root 2041 2009-12-05 06:58 kasp.xml.sample
-rw-r--r-- 1 root root  688 2009-11-07 08:07 zonefetch.xml
-rw-r--r-- 1 root root  948 2009-12-05 06:58 zonefetch.xml.sample
-rw-r--r-- 1 root root  826 2009-11-19 09:45 zonelist.xml
-rw-r--r-- 1 root root  464 2009-12-05 06:58 zonelist.xml.sample
```

Chapitre 2. Les paramètres temporels importants.

Le but de ce chapitre n'est pas de rentrer dans les multiples détails pour la mise en place de DNSSEC. Ce chapitre va tenter d'expliquer comment gérer les clés au travers d'OpenDNSSEC, en liaison avec les fichiers de configuration et un fichier de zone.

Contrairement à d'autres systèmes de gestion DNSSEC, OpenDNSSEC a choisi de gérer pour chaque zone 2 clés KSK et 2 clés ZSK. Pour chaque paire vous avez, une clé activée (indice 256) et une clé en attente (indice 257). J'aime assez ce procédé de double paires; il évite la gestion fastidieuse du temps de propagation lors d'un Rollover pour au moins une des clés. Est-ce vraiment la procédure pure RFC, je n'en suis pas très persuadé...

Ces paramètres sont modifiables indépendamment pour chaque clé, au travers du fichier kasp.xml, et la balise Standby>1</Standby>. La valeur par défaut est fixé à « 1 »(1 en attente).

Pour *OpenDNSSEC*.

- Resign

  Correspond à l'intervalle entre deux exécutions du programme de signature.

- Refresh

  Correspond à un intervalle de rafraîchissement pour les signatures. En général ce délai est plus grand que le temps d'exécution entre deux signatures (Resign). Si aucun changement n'est fait au niveau du « serial », les signatures sont alors générées à nouveau une fois cette limite atteinte.

- Validity

  Correspond à un groupe de deux éléments **Default** et **Denial**. Ces deux éléments sont liés à la dure de vie des signatures.

  Le *`Denial`* est dédié aux signatures NSEC/NSEC3 exclusivement.

  Le *`Default`* est dédié aux valeurs RRSIG exclusivement.

- Jitter

  C'est une durée supplémentaire ajoutée à la date d'expiration de la signature pour s'assurer que toutes les signatures arriveront à expiration au même moment.

- Inception Offset

  Cette valeur est soustraite à l'heure de signature ou d'enregistrement. Elle permet de donner un écart de temps pour diverses opérations (signatures/enregistrements...), afin d'éviter de trouver des signatures dont l'heure de début est supérieure à l'heure réelle.

Pour *`Bind`*.

L'arrivée de DNSSEC bouscule certaines règles purement DNS. En effet avant, il était souhaitable de garder le plus longtemps possible des données en cache. Cette action optimise le réseau en limitant les flux et les requêtes. Avec désormais des clés qui peuvent changer sans préavis, les contraintes ne sont plus pareilles. Il faut être sur de pouvoir garder les informations DNS le plus longtemps possible, sans pour cela compromettre les aspects de sécurité. Ceci reste une analyse personnelle, mais seules des valeurs temporelles relativement courtes permettent de pouvoir recharger une zone (DNSSEC), ou des données pour un DNS rapidement. Néanmoins, tout ceci dépend des données de la zone concernée et de ses modifications, plus ou moins régulières. Je cherche une formule mathématique pour cela... Voici les données (pour mémoire je l'espère) des entêtes à régler au plus juste pour une zone. Je passe sur l'explication des réglages des valeurs *`$TTL`*. Elles aussi doivent respecter la philosophie globale « DNSSEC ».

- Refresh :

  C'est la durée avant vérification de la zone par un serveur secondaire(s). Une petite valeur surcharge inutilement le primaire. Cette valeur doit être relativement petite, car le primaire lors d'un changement (serial = modifié !), notifie de lui-même les secondaires. Voici les paramètres recommandés par le NIST [28] :

- De 20 minutes à 2 heures. Cette valeur doit être plus petite que la durée de vie du RRSIG.

- Retry :

  C'est la durée pour le prochain test de connexion suite à une erreur de connexion, cette valeur ne concerne que les serveurs secondaires. Voici les paramètres recommandés par le NIST [28] :

  - De 05 minutes 01 heures.
  - A noter : D'après le NIST [28], cette valeur devrait correspondre au 1/10 du « REFRESH ».

- Expire :

  Si le serveur « esclave » ne parvient pas à contacter le serveur « maître » , celui-ci va ignorer tout simplement cette zone dès cette valeur d'expiration atteinte. Cette valeur dépend de la fréquence de mise à jour des zones.

  - Une valeur comprise entre 1 et 4 semaines est raisonnable.

- Negative :

  Représente le temps que Bind va garder en mémoire cache (négative) une donnée négative. Comme pour les autres paramètres, il ne faut pas de grande valeur. Des valeurs comprises entre 0 et 10 minutes sont hautement déconseillées. Cette valeur est la référence pour des paramètres DNSSEC, comme les signatures... Voici les recommandations du NIST [28] :

  - une valeur comprise entre 30 secondes et 24 heures est raisonnable.

Note

Si OpenDNSSEC intervient sur une zone, le *`serial`* est automatiquement inscrit, et mis à jour. Par défaut, il correspond au temps, il est au format « unixtime » par défaut.

Les zones à gérer

OpenDNSSEC a été pour moi un sujet d'expérience et non un besoin. Pour mémoire ma zone de test est totalement virtuelle (non routable sur internet). Dans ce document, je vais travailler dans mes exemples uniquement avec la zone de type « FORWARD ». Néanmoins ma « REVERSE » est gérée par OpenDNSSEC, la méthode reste identique pour les deux.

Voici ma zone « vierge » ou non signée :

```
$TTL 2H
@               SOA     portable.archi.amt.  dnsmaster.archi.amt. (
                        1259995801 ; serial (date +%s)
                        1800S      ; refresh
                        2H         ; retry
                        1W         ; expiry
                        1H )       ; minimum
@               NS   portable.archi.amt.
@               NS   serveur.archi.amt.
serveur         A    192.168.1.11
archi.amt.      MX 10 serveur.archi.amt.
www             CNAME        serveur.archi.amt.
wpad            CNAME        serveur.archi.amt.
ftp             A    192.168.1.11
portable2       A    192.168.1.13
portable        A    192.168.1.203
```

Cette zone ne comporte que peu de données, elle reste simple. Les mots *`IN`* peuvent être omis sans problème; le résultat final est identique. Ce qui est changé c'est le *`serial`*. Il n'est plus basé sur la formule classique *`AnneMoisjour000`*, mais sur les secondes écoulées depuis 1970. En effet, cette formule est utilisée par OpenDNSSEC, ce paramètre est modifiable au travers du fichier kasp.xml, dans la rubrique *`Zone`*. C'est une autre façon de faire, qui ne pose aucun problème à Bind également.

Note

Le NIST [28] déconseille d'utiliser pour un DNS public, sans une activation des « view » adaptées, les types suivants : hinfo, txt, loc... Ces mesures peuvent permettre de ne pas trop divulguer d'informations, en vue de limiter les attaques éventuellement...

Chapitre 3. Les fichiers de configuration.

Les divers fichiers de configuration qui suivent servent à OpenDNSSEC pour la gestion des zones et des clés. Tous ces paramètres doivent être adaptés finement à vos besoins et à vos zones.

Note

Les fichiers listés ci-après sont les versions originales. Ils ont fait l'objet du minimum requis pour rendre le système opérationnel.

Le fichier *conf.xml*

Ce fichier est situé dans le répertoire */etc/opendnssec/*(etc-fichiers-conf). Vous devez l'adapter à vos besoins, et remplacer les « # » par votre mot de passe entrée précédemment durant l'initialisation du *slot 0*. L'autre *slot 1(pour moi)*, représente la partie stockage pour mon HSM (via SoftHSM). Compte tenu des clés privées qui y sont stockées, le code « PIN », n'est pas inscrit. Rien de bien particulier pour ce fichier, le paramétrage est clair et simple. Le passage de *Sqlite3* à *Mysql* est réalisé par ce fichier pour OpenDNSSEC.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: conf.xml.in 2056 2009-10-02 09:01:16Z jakob $ -->
<Configuration>
 <RepositoryList>
  <Repository name="softHSM">
   <Module>/usr/local/lib/libsofthsm.so</Module>
   <TokenLabel>OpenDNSSEC</TokenLabel>
   <PIN>##############</PIN>
  </Repository>

 </RepositoryList>
 <Common>
  <Logging>
   <Syslog><Facility>local0</Facility></Syslog>
  </Logging>

  <PolicyFile>/etc/opendnssec/kasp.xml</PolicyFile>
  <ZoneListFile>/etc/opendnssec/zonelist.xml</ZoneListFile>
 <!--
  <ZoneFetchFile>/etc/opendnssec/zonefetch.xml</ZoneFetchFile>
 -->
 </Common>
 <Enforcer>
<!--
  <Privileges>
   <User>opendnssec</User>
   <Group>opendnssec</Group>
  </Privileges>
-->
  <Datastore><SQLite>/var/opendnssec/kasp.db</SQLite></Datastore>
  <Interval>PT3600S</Interval>
 </Enforcer>
 <Signer>
<!--
  <Privileges>
   <User>opendnssec</User>
   <Group>opendnssec</Group>
   <Directory>/</Directory>
  </Privileges>
-->
  <WorkingDirectory>/var/opendnssec/tmp</WorkingDirectory>
  <WorkerThreads>8</WorkerThreads>
  <!-- the <NotifyCommmand> will expand the following variables:
       %zone      the name of the zone that was signed
       %zonefile  the filename of the signed zone
```

```
  -->
     <!-- Cette ligne  t adapt  mes besoins vis-a-vis de <xref linkend="Bind"/> -->
  <NotifyCommand>rndc notify %zone; rndc reload %zone</NotifyCommand>
 </Signer>

 <Auditor>
<!--
  <Privileges>
   <User>opendnssec</User>
   <Group>opendnssec</Group>
   <Directory>/</Directory>
  </Privileges>
-->
  <WorkingDirectory>/var/opendnssec/tmp</WorkingDirectory>
 </Auditor>

</Configuration>a
```

Le fichier *kasp.xml*

Ce fichier permet de configurer tout ce qui concerne les paramètres NSEC3 (ou DNSSEC), en fonction d'une politique. Il est possible de créer plusieurs politiques pour la gestion des zones, une zone ne peut pas être gérée par plusieurs politiques. A noter : toutes les données temporelles doivent commencer par la lettre *P*. Elles doivent être suivies ensuite par la ou les valeurs souhaitées (année, mois, jour, heure et seconde). Elles peuvent être combinées comme ceci par exemple : *P1Y30D5H*.

Pour comprendre les différents paramètres, il est nécessaire de bien connaître DNSSEC/NSEC3.

Ce fichier possède deux champs similaires à celui ci : <Algorithm length="2048">7</Algorithm>. Ces 2 valeurs expriment pour la 1ère (2048), le nombre de bits pour l'encodage des clés de chiffrement (KSK). La 2eme valeur (7) désigne le type d'algorithme à utiliser. A ce jour, le NIST [28] préconise des valeurs au moins égales à 1024 pour l'encodage. Les valeurs de *2048* pour une KSK, et *1024* pour ZSK sont parfaites, ou suffisantes de nos jours.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: kasp.xml.in 2238 2009-10-16 07:38:12Z rb $ -->
<KASP>

 <Policy name="default">
  <Description>A default policy that will amaze you and your friends</Description>
  <Signatures>
   <Resign>PT2H</Resign>
   <Refresh>P3D</Refresh>
   <Validity>
     <Default>P7D</Default>
     <Denial>P7D</Denial>
   </Validity>
   <Jitter>PT12H</Jitter>
   <InceptionOffset>PT300S</InceptionOffset>
  </Signatures>

  <Denial>
   <NSEC3>
    <OptOut/>
    <Resalt>P100D</Resalt>
    <Hash>
     <Algorithm>1</Algorithm>
     <Iterations>5</Iterations>
     <Salt length="8"/>
    </Hash>
   </NSEC3>
  </Denial>

  <Keys>
   <!-- Parameters for both KSK and ZSK -->
   <TTL>PT3600S</TTL>
   <RetireSafety>PT3600S</RetireSafety>
   <PublishSafety>PT3600S</PublishSafety>
```

```
    <!-- <ShareKeys/> -->
    <Purge>P7D</Purge>

    <!-- Parameters for KSK only -->
    <KSK>
     <Algorithm length="2048">7</Algorithm>
     <Lifetime>P365D</Lifetime>
     <Repository>softHSM</Repository>
     <Standby>1</Standby>
     <!-- <ManualRollover/> -->
    </KSK>

    <!-- Parameters for ZSK only -->
    <ZSK>
     <Algorithm length="1024">7</Algorithm>
     <Lifetime>P30D</Lifetime>
     <Repository>softHSM</Repository>
     <Standby>1</Standby>
    </ZSK>
   </Keys>

   <Zone>
    <PropagationDelay>PT9999S</PropagationDelay>
    <SOA>
     <TTL>PT3600S</TTL>
     <Minimum>PT3600S</Minimum>
     <Serial>unixtime</Serial>
    </SOA>
   </Zone>

   <Parent>
    <PropagationDelay>PT9999S</PropagationDelay>
    <DS>
     <TTL>PT3600S</TTL>
    </DS>
    <SOA>
     <TTL>PT3600S</TTL>
     <Minimum>PT3600S</Minimum>
    </SOA>
   </Parent>

   <Audit/>
  </Policy>
</KASP>
```

Ce fichier incorpore 2 paramètres spécifiques à NSEC3. Ses valeurs évitent les attaques par dictionnaire, en créant des valeurs aléatoires. Elles interviennent pour le hachage des signatures. Elles sont réalisées selon un nombre d'itérations (« <Iterations>5</Iterations> »), et de lettres et de chiffres aléatoires. C'est le fameux « salt » (« <Salt length="8"/> »).

Le fichier *zonefecth.xml*

Ce fichier de configuration concerne les paramètres globaux pour OpenDNSSEC vis-à-vis du serveur DNS déclaré en primaire. On retrouve donc les paramètres, comme le port d'écoute, des paramètres non activés pour TSIG, et la déclaration de l'adresse IP du DNS primaire concernée par les zones. Au moins pour le moment, OpenDNSSEC dialogue uniquement avec un seul DNS (primaire).

```
        <?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: zonefetch.xml.in 1920 2009-09-30 07:49:39Z matthijs $ -->
<ZoneFetch>
 <!-- where to listen for notifies -->
 <!-- DEFAULT: do not listen to notify on specific address -->
 <NotifyListen><Port>53</Port></NotifyListen>

 <!-- default inbound AXFR settings
      (per zone setting not yet implemented) -->
```

```
 <Default>
  <!-- TSIG secret for inbound AXFR -->
  <!-- DEFAULT: don't use TSIG -->

  <!-- address of host to request AXFR from -->
  <!-- incoming NOTIFY has to match this address as well -->
  <!-- DEFAULT: none -->
  <RequestTransfer>
   <IPv4>192.168.1.11</IPv4><Port>53</Port>
  </RequestTransfer>
 </Default>
</ZoneFetch>
```

Le fichier *zonelist.xml*

Ce fichier est créé à la suite d'une demande d'intégration de zone. Cette action est réalisée par la commande « ods-ksmutil setup ». Bien que non testé récemment (mais lors de mes essais), OpenDNSSEC ajoutait systématiquement un « pavé » lors d'une demande d'intégration. Je ne peux que conseiller, sans mettre à défaut OpenDNSSEC tout de même, de vérifier ce fichier avant de procéder à la première signature d'une nouvelle zone.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: zonelist.xml.in 1443 2009-07-30 13:17:16Z rb $ -->
<ZoneList>
 <Zone name="archi.amt">
  <Policy>default</Policy>
  <SignerConfiguration>/var/opendnssec/signconf/archi.amt.xml</SignerConfiguration>
  <Adapters>
   <Input>
    <File>/var/opendnssec/unsigned/archi.amt</File>
   </Input>
   <Output>
    <File>/var/opendnssec/signed/archi.amt.signed</File>
   </Output>
  </Adapters>
 </Zone>
 <Zone name="1.168.192.in-addr.arpa">
  <Policy>default</Policy>
  <SignerConfiguration>/var/opendnssec/signconf/1.168.192.in-addr.arpa.xml</SignerConfig
  <Adapters>
   <Input>
    <File>/var/opendnssec/unsigned/1.168.192.in-addr.arpa</File>
   </Input>
   <Output>
    <File>/var/opendnssec/signed/1.168.192.in-addr.arpa.signed</File>
   </Output>
  </Adapters>
 </Zone>
</ZoneList>
```

Chapitre 4. Avant la configuration.

Création du compte « opendnssec »

C'est vraiment pas une obligation, les développeurs ont tout de même tout prévu à cet effet. Ce compte ne me sert à rien chez moi; c'est probablement une erreur que je vais résoudre un jour... Sinon, vous êtes libre de donner un répertoire diffèrent de mon exemple.

adduser --disabled-login --disabled-password --home /var/opendnssec opendnssec

Présentation succincte de mes machines.

J'ai 2 machines différentes avec pour chacune un serveur Bind installé, et OpenDNSSEC pour le DNS primaire, lequel met à jour sur l'autre machine les zones esclaves. C'est un schéma, très simple. La zone "archi.amt" (non routable sur internet) est liée à une plage d'adresse IP en 192.168.x.x. Je suis obligé de présenter ces deux machines, car elles font partie des nombreux exemples dans ce document.

Chapitre 5. TSIG et NSEC/NSEC3

TSIG permet de sécuriser les transferts de zone entre des DNS (AXFR/IFXR). Dans un contexte opérationnel, cette option doit être utilisée pour les dialogues AXFR/IXFR. Cette même option est présente, mais OpenDNSSEC ne l'active pas par défaut. Effectivement, ceci reste une fonctionnalité purement DNS pour moi , et Bind le fait très bien.

Les spécificités DNS

NSEC3 apporte un lot assez conséquent de nouveaux paramètres dans une zone DNS signée. Il est important de les connaître pour une meilleure compréhension. NSEC3 correspond à la version 3 de NSEC. Le problème très connu de NSEC dans sa première version était de pouvoir lister une zone. Et donc de ce fait, découvrir les zones gérées par un serveur. NSEC3 se devait de résoudre ce problème; c'est chose faite désormais. NSEC3 incorpore un RR set [16] spécifique qui incorpore les RR [16] présents dans une zone. En prenant l'exemple ci-dessous, je ne peux interroger cette zone que sur les RR inscrits en gras. Les autres demandes seront rejetées avec le statut « NXDOMAIN », ou sans réponse (ANSWER: 0).

```
IN NSEC3 1 1 5 0AE3245D9DE4A750 5K9E753DH75F9DKQH0JGRGFP8CC807TE NS SOA MX RRSIG DNSKEY
```

Il faut noter aussi que tous les paramètres NSEC3 possèdent comme *TTL* la valeur *SOA* du cache négatif. Il faut aussi noter la présence du RR *NSEC3PARAM* . Ce paramètre est unique dans une zone; il contient des données sur l'algorithme, le ou les drapeaux (0), l'itération et le fameux « salt » pour terminer.

Les « RR » :

Existe depuis longtemps dans le DNS c'est une déclaration de ce style, qui désigne un enregistrement de ressource bien identifiée :

```
ftp  IN A  192.168.1.11
```

Le type d'enregistrement de ressource est très varié; voici une liste très claire (GB) : List_of_DNS_record_types [http://en.wikipedia.org/wiki/List_of_DNS_record_types]

Les « RR set » :

C'est la prise en compte d'enregistrements de ressources du même type (RR). Dans ma zone, le RR set [16] pour le type « A », concerne 3 valeurs. Ces 3 valeurs formeront un seul « RRset ».

```
ftp  IN A  192.168.1.11
portable2 IN A 192.168.1.13
portable IN A 192.168.1.203
```

Les « RR Key » :

Correspond à un paramètre purement DNSSEC, c'est une déclaration de clé publique dans une zone, exemple :

```
archi.amt.  IN DNSKEY 256 3 7 (
    AwEAAZzDvAx3mXnUpIg1nYPq6FF6ezBim2d17vH8VZh+
    pQ/kJUkGODrFftAD82/u53UV2dNiNSxiL9kMNnhrjFJl
    EzbhWVOhhbbTv5Ht0oVWyFnFlqJBt+1CESn04/KPwVEH
    Yiw087hpJ2aTyUte/pfOxvsLP5dwOo71gDLQ6UbS5/et
    ) ; key id = 9783
```

Le chiffre *256* identifie le type de clé (256 = ZSK et 257 = KSK), le chiffre *3* désigne un protocole. Et le chiffre *7* identifie l'algorithme utilisé pour le chiffrement.

Les « RR Sig » :

A chaque *RR Sig* rencontré correspond un RR set [16]. Ils sont donc très présents dans une zone, et se présentent sous cette forme :

```
RRSIG A 7 3 7200 20091230153059 (
 20091223081636 63770 archi.amt.
 b/zpRUwP7Qa/8aeQAnq5T+wUYJfh/hIi2qVL5BHPC2//
 xju0GDM+AGrEdw1kZR0jIookDIpW2Bn/ZAVJ5ksVVLPz
```

```
ddG3aeKJ2KyHd+UerbK54YRP8iOI3Q5Zm2hWbNnvu8fp
+VlIuxhoSULOJK+R8007a4jcHwEz316boWfrHNU= )
```

Les valeurs suivantes sont intéressantes à connaître. La lette « A » désigne un RR set [16] de type « A ». La valeur « 7 » désigne un encodage de type « RSASHA1-NSEC3-SHA1 », ce chiffre est un alias du chiffre 5 (très utilisé et rencontré). Le chiffre « 3 » représente le nombre de labels. Exemple : "www.archi.amt" à 3 labels (www + archi + amt), pour une racine le label est égal à zéro. Le chiffre « 7200 » est exprimé en secondes et correspond à la valeur du TTL. Ensuite suivent la date d'expiration puis la date de création de cette signature, à la seconde près. Le chiffre *63770* correspond à un numéro de clé, la KSK active précisément. Les mots (« archi.amt. ».) représente la zone concernée, ensuite c'est la signature sur plusieurs lignes (Bind est limité en longueur de caractères dans une zone).

Le mot *Rdata* est parfois utilisé. Il représente tous les enregistrements RRSIG, à l'exception des clés.

Les « DS » :

C'est une spécificité de DNSSEC. Il correspond au terme anglais : « Delegation Signer ». Ce paramètre permet de faire authentifier une zone fille vers une zone parente. Le paragraphe suivant (Délégation-signature [25]) explique « en détail » cette procédure.

Chapitre 6. Configuration et exploitation.

Comme déjà annoncé, la zone en exemple est totalement fictive et non reconnue sur internet. Néanmoins « OpenDNSSEC » ne le devine pas, et ceci permet de se faire la main sur un domaine et sa racine sans contrainte extérieure. Comme déjà présentée, voici ma zone de référence, qui reste très basique : archi.amt. Cette zone, ou plutôt ce fichier doit être placé dans le répertoire « *unsigned* », dont la base est pour moi */var/opendnssec*. Il va de soi que les fichiers de configuration doivent être prêts pour la prise en compte de cette zone. Il s'agit des fichiers présents dans le répertoire */etc/opendnssec*, mais en réalité c'est l'ensemble d'OpenDNSSEC qui est concerné.

Création des bases de données.

La création du premier slot (BD) est obligatoire pour le bon fonctionnement d'OpenDNSSEC. Le deuxième slot (slot 1) est obligatoire si vous n'utilisez pas un HSM. C'est mon cas. A noter que l'exemple propose le slot 0 et le slot 4. Personnellement, j'ai fait un slot « 0 » et un slot « 1 ». Je ne rencontre aucun problème avec cette solution. Il faut éditer le fichier softhsm.conf pour qu'il ressemble à une version similaire.

```
# softHSM configuration file
#
0:/var/opendnssec/slot0.db
1:/var/opendnssec/kasp.db
```

Il faut prévoir d'utiliser Sqlite3, et donc, il faut pour cela initialiser les données nécessaires.

```
cd /var/opendnssec/ ; softhsm --init-token --slot 0 --label "OpenDNSSEC"

softhsm --init-token --slot 1 --label "kasp.db"
```

Note

Le mot (exemple) « OpenDNSSEC » doit être présent dans le fichier conf.xml, dans la rubrique *softHSM* (au début). C'est impératif, et pour le 2eme slot (slot 1), il doit être indiqué dans la rubrique *Enforcer*, dans le même fichier. Retenez bien les mots de passe (PIN). Vous êtes libre de donner un nom de label comme il vous convient.

```
softhsm --init-token --slot 0 --label "OpenDNSSEC"
The SO PIN must have a length between 4 and 255 characters.
Enter SO PIN:########
The user PIN must have a length between 4 and 255 characters.
Enter user PIN:########
The token has been initialized.

# /var/opendnssec# softhsm --init-token --slot 1 --label "kasp.db"
The SO PIN must have a length between 4 and 255 characters.
Enter SO PIN:############
The user PIN must have a length between 4 and 255 characters.
Enter user PIN:############
The token has been initialized.


root@serveur:/var/opendnssec# ls -l
-rw-r--r-- 1 root root 5120 2009-12-06 17:10 slot0.db
-rw-r--r-- 1 root root 5120 2009-12-06 17:10 kasp.db
```

Si problème avec les BD.

Avant que OpenDNSSEC ne soit stable, à ses débuts il était régulier lors d'une installation d'intervenir sur les bases de données. Ce problème semble inexistant depuis plusieurs semaines. Désormais, si jamais le message suivant apparaît, c'est que vous avez pas ou mal configuré l'un des « slot »:

```
Dec 16 07:30:16 serveur ods-ksmutil: SoftHSM: init: Wrong database schema version: /var/
```

Il suffit pour cela de rechercher dans les sources de OpenDNSSEC, le fichier *database_create.sqlite3* (opendnssec-1.0.0rc1/enforcer/utils/database_create.sqlite3). Une fois localisé, placez vous dans le répertoire des

BD (/var/opendnssec/) puis faire cette commande pour la base de données concernée (slot). Le fait d'exécuter cette commande sur les 2 slots ne pose pas de problème, ceci apporte une structure SQL plus grosse dans les BD, pas forcément utilisées après.

```
sqlite3 kasp.db < /tmp/database_create.sqlite3
```

La taille du fichier pour chaque base de données doit évoluer dans le temps.

Voici une commande pour lister les données d'une base :

```
echo ".dump" | sqlite3 slot0.db
```

Prise en compte de la zone (tape initiale).

Note

A partir de maintenant, je ne peux que vous conseiller de laisser un terminal avec un « *tail -f /var/log/syslog* ». Cette action vous permettra de suivre en directe les évolutions.

Voici la première commande à faire :

```
# ods-ksmutil setup
*WARNING* This will erase all data in the database; are you sure? [y/N] y
SQLite database set to: /var/opendnssec/kasp.db
fixing permissions on file /var/opendnssec/kasp.db
zonelist filename set to /etc/opendnssec/zonelist.xml.
kasp filename set to /etc/opendnssec/kasp.xml.
Repository softHSM found
No Maximum Capacity set.
RequireBackup NOT set; please make sure that you know the potential problems of using ke
Policy default found
Zone archi.amt found
Policy set to default.
Added zone archi.amt to database
Zone 1.168.192.in-addr.arpa found
Policy set to default.
Added zone 1.168.192.in-addr.arpa to database
```

Votre zone est donc prise en compte par OpenDNSSEC, mais c'est tout. En réalité cette commande permet d'ajouter une zone ou plusieurs zones, elle se comporte comme une initialisation. Cette commande est donc potentiellement dangereuse si non maîtrisée.

Il faut démarrer désormais OpenDNSSEC avec en particulier ses deux programmes résidents. Voici la commande à réaliser :

```
ods-control start
Starting signer engine...
connecting to /var/run/opendnssec/engine.sock
OpenDNSSEC signer engine version 1.0.0rc1
Zone list updated: 0 removed, 2 added, 0 updated
running as pid 1611
Starting enforcer...
OpenDNSSEC ods-enforcerd started (version 1.0.0rc1), pid 1613>
```

Dès le lancement, et surtout pour la première fois, il est important de regarder les très nombreuses lignes écrites dans le fichier */var/log/syslog*. Dans le même répertoire, vous pouvez regarder également le fichier *debug*. Malgré le nombre de lignes écrites, ces traces sont très lisibles.

Note

OpenDNSSEC ne gère les clés que par le biais des bases de données, ou d'un HSM. Il est nécessaire de sauvegarder les clés par une sauvegarde de celles ci. Voici la commande à réaliser :

```
ods-ksmutil backup done
SQLite database set to: /var/opendnssec/kasp.db
Marked all repositories as backed up at 2009-12-23 14:30:29
```

Cette commande est différente de la sauvegarde des bases de données (*database backup*).

Les deux programmes résidents sont ici, pour information

```
root  1611  0.1  0.4  85544 13772 ?  Sl  06:44  0:00 /usr/bin/python /usr/local/lib/open
root  1613  0.0  0.1  17360  4456 ?  Ss  06:44  0:00 /usr/local/sbin/ods-enforcerd>
```

En théorie les programmes résidents s'activent immédiatement. Et si le besoin est avéré, la création des signatures s'opère, ou les mises jour. Néanmoins si cette procédure ne se lance pas, après recherche d'une possible erreur, un forçage est possible avec cette commande :

```
ods-ksmutil key generate --interval 1 --policy default
```

Auditer une zone.

Pour avoir essayer, les développeurs déconseillent l'emploi de la commande suivante. Néanmoins, elle peut servir tout de même pour en particulier vérifier si une erreur existe dans votre zone.

```
/usr/local/bin/ods-auditor -c /etc/opendnssec/conf.xml -s /var/opendnssec/tmp/archi.amt.
Auditor started
Auditor starting on archi.amt
6: SOA differs : from 2009120901 to 1261115043
6: Auditing archi.amt zone : NSEC3 SIGNED
6: Finished auditing archi.amt zone
Auditor found no errors
```

La zone signée.

Voici ce que donne une petite zone signée. C'est tout de suite moins lisible, mais la zone est désormais sécurisée, différent d'authentifiée (!). OpenDNSSEC apporte son lot d'informations, ou de nouveautés dans une zone signée. Je parle en tant que fichier et non en tant que zone chargée au niveau DNS. Il est donc possible de lire la date et l'heure de la dernière mise à jour. Ceci vite de scruter les « logs » pour vérifier ces informations. Et la fin de la zone, OpenDNSSEC indique les modifications apportées à la zone lors de cette mise à jour. Certes c'est peu important. Chaque clé est aussi clairement identifiée, avec pour chacune un commentaire. Dans l'exemple suivant, les clés et les signatures ont été réduites volontairement.

```
; Signed on 2010-01-18 19:19:32
archi.amt. 7200 IN SOA portable.archi.amt. dnsmaster.archi.amt. 1263838772 86400 14400 6
wpad.archi.amt. 43200 IN CNAME serveur.archi.amt.
wpad.archi.amt. 43200 IN RRSIG CNAME 7 3 43200 20100125175153 20100118061710 63770 archi
3rm8965p1rfpq172a4kvi1g0rs8t8jnr.archi.amt. 3600 IN NSEC3 1 1 5 0ae3245d9de4a750  3uld39
3rm8965p1rfpq172a4kvi1g0rs8t8jnr.archi.amt. 3600 IN RRSIG NSEC3 7 3 3600 20100125165656
archi.amt. 7200 IN NS serveur.archi.amt.
archi.amt. 7200 IN NS portable.archi.amt.
archi.amt. 7200 IN RRSIG NS 7 2 7200 20100125071507 20100118061710 63770 archi.amt. fae2
archi.amt. 7200 IN RRSIG SOA 7 2 7200 20100126054915 20100118181432 63770 archi.amt. Hb/
archi.amt. 43200 IN MX 10 serveur.archi.amt.
archi.amt. 43200 IN RRSIG MX 7 2 43200 20100125091203 20100118061710 63770 archi.amt. Fd
archi.amt. 7200 IN DNSKEY 256 3 7 AwEAAZzDvAx3mXnUp...Q6UbS5/et ;{id = 9783 (zsk), size
archi.amt. 7200 IN DNSKEY 256 3 7 AwEAAfJGDaBiwiwHx3s...fsdR6H8Nd ;{id = 63770 (zsk), si
archi.amt. 7200 IN DNSKEY 257 3 7 AwEAAb+1wO0r9fLWJf...+Kig9oG0= ;{id = 31939 (ksk), siz
archi.amt. 7200 IN DNSKEY 257 3 7 AwEAAfJGujScAu2...1pEzYSnrHlGM= ;{id = 31573 (ksk), si
archi.amt. 7200 IN RRSIG DNSKEY 7 2 7200 20100125102105 20100118061710 31939 archi.amt.
archi.amt. 3600 IN NSEC3PARAM 1 0 5 0ae3245d9de4a750
archi.amt. 3600 IN RRSIG NSEC3PARAM 7 2 3600 20100125063156 20100118061710 63770 archi.a
3uld3901jg68fbd092cm3pekrdtobqet.archi.amt. 3600 IN NSEC3 1 1 5 0ae3245d9de4a750  5k9e75
3uld3901jg68fbd092cm3pekrdtobqet.archi.amt. 3600 IN RRSIG NSEC3 7 3 3600 20100125171022
serveur.archi.amt. 7200 IN A 192.168.1.11
serveur.archi.amt. 7200 IN RRSIG A 7 3 7200 20100125092619 20100118061710 63770 archi.am
5k9e753dh75f9dkqh0jgrgfp8cc807te.archi.amt. 3600 IN NSEC3 1 1 5 0ae3245d9de4a750  769ivr
5k9e753dh75f9dkqh0jgrgfp8cc807te.archi.amt. 3600 IN RRSIG NSEC3 7 3 3600 20100126045418
*.archi.amt. 7200 IN A 192.168.1.11
*.archi.amt. 7200 IN RRSIG A 7 3 7200 20100125191229 20100118181432 63770 archi.amt. GrQ
769ivrbvqnqm8s6rhl5vq5c80cjaid9r.archi.amt. 3600 IN NSEC3 1 1 5 0ae3245d9de4a750  bvsm6k
769ivrbvqnqm8s6rhl5vq5c80cjaid9r.archi.amt. 3600 IN RRSIG NSEC3 7 3 3600 20100126022127
portable2.archi.amt. 7200 IN A 192.168.1.13
portable2.archi.amt. 7200 IN RRSIG A 7 3 7200 20100125124812 20100118061710 63770 archi.
bvsm6k878jcadruks6ac9qv7ihofa99b.archi.amt. 3600 IN NSEC3 1 1 5 0ae3245d9de4a750  e0530g
bvsm6k878jcadruks6ac9qv7ihofa99b.archi.amt. 3600 IN RRSIG NSEC3 7 3 3600 20100125102917
dlv.archi.amt. 0 IN TXT "DLV:1:eruerfdegsdd"
dlv.archi.amt. 0 IN TXT "DLV:1:neaintftitim"
dlv.archi.amt. 0 IN RRSIG TXT 7 3 0 20100125124700 20100118061710 63770 archi.amt. vGSlc
```

```
e0530gb2636076nie11asvusegvna1rk.archi.amt. 3600 IN NSEC3 1 1 5 0ae3245d9de4a750  jk83ut
e0530gb2636076nie11asvusegvna1rk.archi.amt. 3600 IN RRSIG NSEC3 7 3 3600 20100125095629
www.archi.amt. 43200 IN CNAME serveur.archi.amt.
www.archi.amt. 43200 IN RRSIG CNAME 7 3 43200 20100125133133 20100118061710 63770 archi.
jk83utn4ffjckv1io3a0cl8rn2aardoj.archi.amt. 3600 IN NSEC3 1 1 5 0ae3245d9de4a750  leef2v
jk83utn4ffjckv1io3a0cl8rn2aardoj.archi.amt. 3600 IN RRSIG NSEC3 7 3 3600 20100125100056
ftp.archi.amt. 7200 IN A 192.168.1.11
ftp.archi.amt. 7200 IN RRSIG A 7 3 7200 20100125111110 20100118061710 63770 archi.amt. T
leef2ve7akn9ra43ua1n420olvfcvcfi.archi.amt. 3600 IN NSEC3 1 1 5 0ae3245d9de4a750  pe0dg5
leef2ve7akn9ra43ua1n420olvfcvcfi.archi.amt. 3600 IN RRSIG NSEC3 7 3 3600 20100125063916
portable.archi.amt. 7200 IN A 192.168.1.203
portable.archi.amt. 7200 IN RRSIG A 7 3 7200 20100125073502 20100118061710 63770 archi.a
pe0dg5rfj07sqamjjuo0mpsgfq8g658i.archi.amt. 3600 IN NSEC3 1 1 5 0ae3245d9de4a750  3rm896
pe0dg5rfj07sqamjjuo0mpsgfq8g658i.archi.amt. 3600 IN RRSIG NSEC3 7 3 3600 20100125132426
; Last refresh stats: existing: 19, removed 1, created 3
```

Voici ce qu'il faut retenir de tout cela :

La clé KSK activée possède le numéro *31939*, car le seul "RRSIG DNSKEY" inclut également ce numéro. La ZSK activée porte le numéro *63770*, ce chiffre est régulièrement inscrit dans les paramètres NSEC3 (signature et clé). On retrouve également les clés « activées » et « en attente ». Les clés KSK sont identifiables, car elles comportent les mots *DNSKEY 256 3 7*, et les ZSK par les mots *DNSKEY 257 3 7*. La taille de chaque clé peut-être prise en compte, comme dans mon exemple, avec le passage de 1024 à 2048 bits pour chaque encodage. Il existe un RR set [16] par type NS, A, MX, CNAME. Le chiffre *7* très souvent présent dans les valeurs DNSSEC représente un encodage de type *RSASHA1-NSEC3-SHA1*.

OpenDNSSEC a également pris en compte le caractère « * », en tant que « poubelle DNS »...

Et pour information, entre le passage clair et signé, ma zone de test a grossi de sept fois.

Problème avec le *serial et effacement*.

Lors de mes premiers essais, j'ai remarqué que OpenDNSSEC prend comme référence un « *serial* » basé sur le temps Unix. Et donc, il y avait incohérence entre les paramètres de mes zones basées sur les valeurs *YYYYYMMJJxxx*. Voici une procédure pour faire « marche arrière »; possible dès les premiers « pas » avec OpenDNSSEC. Il faut arrêter les programmes résidents par cette commande **ods-control stop**, puis demander la suppression de toutes informations dans les bases de données via cette commande : **ods-ksmutil setup**. Effacer tous les fichiers situés dans */var/opendnssec/*, à l'exception de vos fichiers de zone « vierge ». Ensuite relancer les 2 programmes résidents : **ods-control start**. Votre zone est à nouveau signée, avec dans ma situation un *serial* de zone basé sur la date exprimée en secondes depuis 1970. C'est aussi la même valeur utilisée par OpenDNSSEC; mes zones sont cohérentes désormais. Cette valeur (serial) peut être trouvée rapidement par cette commande **date +%s**.

La solution de modifier le « serial » dans la zone non signée est aussi possible (c'est fiable et plus rapide). Voilà tout de même une solution d'éffacement de tout...

Ajouter une zone.

Une fois votre zone « vierge » installée (répertoire *unsigned*), il faut demander sa prise en compte par OpenDNSSEC, par le biais d'une mise à jour. La commande qui suit modifie le fichier zonelist.xml. A ce niveau, vous pouvez également contrôler cette prise en compte au travers de celui-ci. Ensuite vous pouvez exécuter cette commande; votre zone sera prise en compte comme les autres par la suite.

```
ods-ksmutil update zonelist
SQLite database set to: /var/opendnssec/kasp.db
zonelist filename set to /etc/opendnssec/zonelist.xml.
kasp filename set to /etc/opendnssec/kasp.xml.
Zone archi.amt found
Policy set to default.
Zone 1.168.192.in-addr.arpa found
Policy set to default.
```

Il faut par la suite vérifier la prise en compte des créations de signatures, quitte à relancer ods-control si besoin est.

Suppression d'une zone.

Les commandes suivantes peuvent intervenir sur une ou plusieurs zones.

La commande est simple et efficace; elle se passe de commentaire :

```
ods-control --zone delete nom-de-la-zone>
```

, ou pour tout effacer :

```
ods-control zone delete --all
```

Vous pouvez vous référer au chapitre serial et effacement [21] pour contrôler le bon déroulement des actions.

Signer à nouveau et IMMÉDIATEMENT une zone.

Cette commande est utiliser pour une mise jour d'une zone après modification. Il ne faut pas confondre cette commande avec le fameux Rollover, qui intervient sur les KSK ou ZSK.

Les deux programmes résidents en fonction de vos paramètres signent automatiquement la ou les zones, à condition que votre *serial* ait été modifié. Néanmoins, la commande suivante permet de « forcer » la génération de nouvelles signatures uniquement.

```
# ods-signer sign archi.amt
connecting to /var/run/opendnssec/engine.sock
Zone scheduled for immediate resign
```

Autres mises à jour.

Ceci s'adresse plutôt aux très multiples commandes que propose *ods-ksmutil*, mais elles font partie aussi des commandes d'exploitation courantes.

- **ods-ksmutil update kasp**

  Pour une une mise à jour suite à modification dans le fichier kasp.xml.

- **ods-ksmutil update zonelist**

  Pour la prise en compte d'une ou plusieurs zones, suite à modification du fichier *zonelist.xml*.

- **ods-ksmutil update conf**

  Pour une mise à jour portant sur un ou plusieurs fichiers de configuration.

- **ods-ksmutil update all**

  C'est la commande qui fait une mise à jour globale vis-à-vis des fichiers de configuration.

Les roulements de clés ou Rollover .

Il ne faut surtout pas confondre le Rollover avec un rafraîchissement des signatures. En effet, vous pouvez très bien avoir une KSK avec une durée de vie basée sur 3 ans, et renouveler sa signature toutes les semaines ou tous les mois. Dans cette situation le numéro de clé reste inchangé.

Le roulement ou le fameux « Rollover » est géré automatiquement dans un contexte classique de fonctionnement de votre DNS (non testé faute de temps...).

Cette action dite « forcée » doit être réalisée suite à un accident, une compromission ou une suspicion vis-à-vis de vos clés, ou de son stockage (HSM, base de données...). OpenDNSSEC offre cette possibilité de forcer cette procédure manuellement. Cette action automatique peut-être désactivée en activant l'option *<!-- <ManualRollover/> -->* dans le fichier *kasp.xml*.

Une fois de plus OpenDNSSEC gère cette action à merveille : vous n'avez pas grand-chose à faire. A noter que le roulement des clés « forcé » doit rester une action rare, comme écrit dans le RFC 4641(suite du RFC2541). Voici les différences entre un roulement de clé pour une KSK et pour une ZSK :

- La ZSK ne concerne que le primaire, les secondaires et les données en cache.
- La KSK prend en compte les mêmes contraintes que la ZSK, mais doit en plus transmettre sa nouvelle clé au travers de l'enregistrement DS, à sa zone parente.

A ce sujet, les gestionnaires de la zone « .fr » ou « .com » (exemples), devraient presque monter une permanence afin de maintenir les sous zones actualisées. Heureusement pour eux, c'est l'inverse qui se produit avec les fameux « Trust Anchors (RFC 5011) ». C'est la racine qui envoie une clé de confiance, et non l'inverse.

Pour changer la KSK (exemple), il suffit de suivre l'une des procédures suivantes :

Note

Si vous devez « un jour » changer la KSK et la ZSK en même temps, c'est impossible. Il faut commencer le roulement par la ZSK, puis finir par la KSK. Car c'est elle qui valide les actions faites par la ZSK.

Note

Un roulement de clés « dans l'urgence » se déroule très bien avec des paramètres temporels relativement courts.

Selon une politique précise.

Automatiquement OpenDNSSEC va prendre en compte tous les paramètres et générer une nouvelle clé KSK ou ZSK, et selon une politique précise. La commande suivante ne génère une clé KSK que pour une politique précise, donc pour une ou plusieurs zones :

```
ods-ksmutil key rollover --policy default --keytype KSK
```

Pour une zone précise.

La commande est très similaire à la précédente; les principes restent inchangés.

```
ods-ksmutil key rollover --zone archi.amt --keytype KSK
```

Création et publication.

Comme vous pouvez le voir dans la liste qui suit, ma zone contient trois KSK (active / ready / publish), c'est parfait. Après un délai de propagation qui doit être au minimum équivalent au double du TTL, la clé *54868* est prête à être remplacée par le numéro *31939*, laquelle sera remplacée par la *31573*.

```
ods-ksmutil key list --verbose --zone archi.amt
SQLite database set to: /var/opendnssec/kasp.db
Keys:
Zone:       Keytype:  State:    Date of next transition:  CKA_ID:       Repository:   Ke
archi.amt     KSK     active    2010-01-04 12:07:28      44e0d8781f8...  softHSM      5
archi.amt     KSK     ready     next rollover            5a1c7700083...  softHSM      3
archi.amt     ZSK     active    2010-01-22 09:21:36      fc1b9fc2397...  softHSM      6
archi.amt     ZSK     ready     next rollover            6185307cff2...  softHSM
archi.amt     KSK     publish   2010-01-04 18:41:49      7bae1e5d855...  softHSM      3
```

Activation / roulement.

Avant d'activer le roulement, il est nécessaire de vérifier ou de transmettre les paramètres « DS » (Délégation-signature [25]) pour une intégration dans la zone parente. Une fois cette tape terminée, le roulement peut commencer :

```
ods-ksmutil key ksk-roll --zone archi.amt --keytag 31939
*WARNING* This will retire the currently active KSK; are you sure? [y/N] y
SQLite database set to: /var/opendnssec/kasp.db
Found key with CKA_ID 5a1c7700083898f0e3b2af91c26152db
Key 5a1c7700083898f0e3b2af91c26152db made active, old key retired
```

Parfait, la commande est exécutée, et voilà le résultat :

```
ods-ksmutil key list --verbose --zone archi.amt
SQLite database set to: /var/opendnssec/kasp.db
Keys:
Zone:       Keytype:  State:    Date of next transition:  CKA_ID:     Repository:     K
archi.amt     KSK     retire    2010-01-05 15:49:12      44e0d78...  softHSM         5
archi.amt     KSK     active    2011-01-05 09:02:33      5a1c700...  softHSM         3
archi.amt     ZSK     active    2010-01-22 09:21:36      fc1b923...  softHSM         6
archi.amt     ZSK     ready     next rollover            61853f2...  softHSM
archi.amt     KSK     ready     next rollover            7bae198...  softHSM         3
```

Sans même vérifier, j'ai fait également une mise à jour de cette zone, puis transmis les nouveaux paramètres DS (Délégation-signature [25]) à ma zone parente. Cette même zone parente a également été résigné, et pour finir j'ai demandé à Bind de recharger les nouvelles zones par sécurité.

La suite se fait automatiquement en fonction des paramètres inscrits dans votre fichier kasp.xml. Après plusieurs heures, ou même plusieurs jours selon vos paramètres, votre ancienne clé va être remplacée puis supprimée par la nouvelle, et le tout sans rien faire.

Lister les clés

La commande suivante liste l'ensemble des clés qui sont gérées par OpenDNSSEC. Cette commande peut être utilisée à titre de contrôle. La commande adaptée pour lister les clés d'une zone précise :

```
ods-ksmutil key list --zone archi.amt
```

La commande pour lister les clés de toutes les zones (tombe bien y a qu'une zone...) :

```
ods-ksmutil key list --verbose --all
SQLite database set to: /var/opendnssec/kasp.db
Keys:
Zone:       Keytype:  State:    Date of next transition:  CKA_ID:      Repository:   Key
archi.amt   KSK       active    2011-01-05 09:02:33       5a1c770008...  softHSM      31
archi.amt   ZSK       active    2010-01-22 09:21:36       fc19fc2397...  softHSM      63
archi.amt   ZSK       ready     next rollover             618c4fff29...  softHSM       9
archi.amt   KSK       ready     next rollover             7bae5dba98...  softHSM      31
```

Et pour vérifier si cela est vrai, c'est tout de même important, il suffit de « rentrer » dans la zone concernée. De là, vous devez avoir dans votre zone des lignes similaires aux lignes qui suivent. OpenDNSSEC à même inscrit à quoi correspond chaque valeur.

```
archi.amt.      3600    IN      DNSKEY  256 3 7 AwEAADFEz67ba68...D7foMWyr ;{id = 30150
archi.amt.      3600    IN      DNSKEY  256 3 7 AwEAAbz1yXAUYqf...oZLKG6vr ;{id = 41180
archi.amt.      3600    IN      DNSKEY  257 3 7 AwEAAdN3K+WqygF...42XI5VM= ;{id = 17010
archi.amt.      3600    IN      DNSKEY  257 3 7 AwEAAdVjDibBSFy...P+d+Ubk= ;{id = 48021
```

Cette commande est similaire, avec tout de même les explications détaillés sur les clés en moins. Son avantage est de pouvoir questionner à distance :

```
dig +dnssec archi.amt dnskey @127.0.0.1 +multilines
```

Chapitre 7. Délégation de signature.

Pour le moment et dans ce document, je ne parle que d'une seule zone. Cette zone 100% fictive « archi.amt » dépend de sa mère « amt ». DNSSEC suit la philosophie du DNS; cette hiérarchie est basée sur une arborescence. Et l'authentification doit suivre impérativement cette pratique, dans une logique purement scurisée/authentifiée (100% DNSSEC). Bien que DNSSEC et OpenDNSSEC savent très bien gérer une zone « fille » signée, avec une zone parente non signée sans problème(DNSSEC). Il est logique en terme d'authentification que la zone « mère (amt.) » soit signée, mais encore mieux, qu'elle reconnaisse sa zone fille (archi.amt) comme authentifiée. J'insiste sur le mot « logique », car rien n'interdit de sécuriser une zone fille et de laisser sa mère en position non signée (insecure). Cette authentification fille-mère est possible par la délégation de signature au travers de paramètres (DS). OpenDNSSEC à bien évidemment prévu cette situation, et mets à votre disposition une commande spécifique à cet effet.

Création d'un enregistrement *DS*

La commande suivante permet d'extraire d'une zone précise, deux enregistrements de type « DS ». Ces deux valeurs « DS » sont basées sur la KSK active de la zone fille. Ces données sont ensuite à transmettre de façon sécurisée pour une inclusion dans la zone parente. Ce système a été adopté par sa simplicité et son efficacité, sans contrainte pour les autres (TLD,...).

```
ods-ksmutil key export --zone archi.amt --keystate active --ds
SQLite database set to: /var/opendnssec/kasp.db

;active KSK DS record (SHA1):
archi.amt. 3600 IN DS 54868 7 1 03828ba9eba3880782b9c3c5120339bf7696535d
; xebom-didyp-nypyp-fideb-lebur-nebes-hugob-fyvar-zuton-kigyh-texox

;active KSK DS record (SHA256):
archi.amt. 3600 IN DS 54868 7 2 0e59918a88476f7219b86c02798b7e360e42b97705f07807f7c7c47e
; xefih-nigom-podug-loril-dekir-myryb-dyvym-ryzof-kifug-divul-lucyz-bivob-lotus-lycil-vo
```

Faites la même procédure mais avec les clés en attente/prêtes (passage de active à ready). Personnellement je trouve cette procédure étrange, car je ne vois pas la plus value par cet apport de clé « en attente » au niveau supérieur... Néanmoins voici la commande :

```
ods-ksmutil key export --zone archi.amt --keystate ready --ds
```

L'exportation de clés se fait au travers de 2 signatures, elles sont identiques mais seulement encodées différemment (valeurs 1 et 2). Pour les intégrer dans la zone parente, il suffit de retirer la valeur du TTL, d'ajouter des parenthèses pour encadrer chaque signature :

```
$TTL 3600 ; 1 hour
$ORIGIN amt.
archi       DS  54868 7 1 (03828BA9EBA3880782B9C3C5120339BF7696535D )
DS  54868 7 2 (0E59918A88476F7219B86C02798B7E360E42B97705F07807F7C7C47E455FFBA1 )
```

Seul le numéro fait que la clé est valide, pour cet exemple c'est le numéro *54868*. Ce chiffre doit absolument correspondre aux valeurs « DS », mais aussi à la clé active (KSK) de la zone fille.

Et voilà, le tour est joué, c'est très simple, une fois votre serveur DNS est relancé, ou rechargé.

Il reste encore une autre possibilité que je n'ai pas testée. C'est la récupération d'une clé de confiance émise par un parent, ou même par un TLD. C'est le fameux paramètre *trusted-keys* utilisé par Bind. Les articles sur le NET à ce sujet ne manquent pas.

Chapitre 8. Bind et OpenDNSSEC

Une fois la ou les zones signées, voici comment est perçu votre zone au travers de Bind, si celui ci est bien configuré. Les zones sont prises en compte sans problème, elles sont marquées dans les historiques en *(DNSSEC signed)*. La version de Bind utilisé par moi-même est la *9.7.0rc2*, pas encore déclarée comme stable. Néanmoins, je n'ai pas eu de problème avec mes zones signées. Il faut tout même noter ceci, et c'est nouveau pour moi. Bind ajoute pour chaque fichier de zone, un autre fichier qui ressemble à un fichier « journal (.jnl) ». Ces fichiers sont présents à coté des fichiers de zone signés (répertoire « signed »). Suite à des problèmes lors du lancement de Bind, j'ai inclus la suppression de ce type de fichier (*.jnl) dans le script de lancement. La prise en compte d'une zone signée par Bind est suivie par la création de ce journal (ou fichier). A ce propos, le répertoire */var/opendnssec/signed* doit être accessible en écriture pour le *user* de Bind (pas le root!)

```
20-Dec-2009 19:12:36.445 general: info: zone archi.amt/IN: loaded serial 1261332715 (DNS
20-Dec-2009 19:12:36.445 general: info: zone 0.0.127.in-addr.arpa/IN: loaded serial 2008
20-Dec-2009 19:12:36.446 general: info: zone 1.168.192.in-addr.arpa/IN: loaded serial 12
20-Dec-2009 19:12:36.446 general: info: zone localhost/IN: loaded serial 2008010101
20-Dec-2009 19:12:36.446 general: notice: running
20-Dec-2009 19:12:36.447 general: notice: zone archi.amt/IN: setting keywarntime to 1261
20-Dec-2009 19:12:36.447 general: notice: zone 1.168.192.in-addr.arpa/IN: setting keywar
```

Les modifications à apporter à Bind.

L'ensemble des modifications qui vont être décrites, intervient dans le fichier *named.conf*, à l'exception des paramètres pour le traitement des historiques Bind.

Compilation de Bind.

Il est nécessaire d'adapter Bind avec des paramètres spéciaux lors de la compilation, pour pouvoir bénéficier de toutes ses possibilités DNSSEC. Ces paramètres permettent d'utiliser la commande « dig » avec l'option « +sigchase ». La compilation n'est pas une obligation, la plupart des distributions Gnu/Linux fournissent un Bind récent, mais avec très probablement des capacités DNSSEC minimalistes .

```
./configure --prefix=/usr \
--host=i686-pc-linux-gnu -mandir=/usr/share/man \
--build=i686-pc-linux-gnu \
--infodir=/usr/share/info --datadir=/usr/share \
--sysconfdir=/etc/bind --localstatedir=/var \
--with-libtool --with-openssl \
--enable-linux-caps --enable-threads \
--with-randomdev=/dev/urandom \
--enable-threads \
CFLAGS="${CFLAGS} -fno-strict-aliasing -DDIG_SIGCHASE=1" \
&& make && make install
```

Dans la partie *options* (named.conf):

Les paramètres qui suivent sont transmis pour information. Ils ne nécessitent pas de commentaire particulier.

```
dnssec-enable yes;
        dnssec-validation yes;
```

Vous pouvez également ajouter cette option, qui se passe de commentaire : *dnssec-accept-expired yes;*.

Dans la partie *zones* (named.conf)

Il faut aussi indiquer à Bind comment gérer la zone, et où la trouver; ceci reste un exemple :

```
zone "archi.amt" IN {
type master;
file "/var/opendnssec/signed/archi.amt.signed";
allow-transfer { ADMINS; 192.168.1.203; };
notify yes;
};
```

Chapitre 9. Conclusion.

L'année 2010 devrait être importante en ce qui concerne le déploiement de DNSSEC. En effet, il est important de sécuriser le réseau DNS sur INTERNET, pour en particulier avoir la certitude d'être sur le bon site, la bonne adresse IP. Après plusieurs années (mars 2003) pour la mise au point des paramètres et des évolutions, petit à petit les requêtes DNS, les « résolveurs », et par la suite même votre navigateur parlera « DNSSEC ». Et comme pour le protocole HTTPS, la moindre incohérence fera l'objet d'un avertissement instantanément. Mais actuellement, DNSSEC « bouge » encore (RFC, Draft), et ne devrait pas se stabiliser avant plusieurs années. Ceci ne remet pas en cause son emploi, il peut seulement évoluer ou s'adapter dans le temps. A noter aussi, les dernières versions de Bind incluent désormais un support, ou plutôt une gestion complète DNSSEC. Au niveau logiciel, et au plus près du client, après maintes recherches je ne trouve pas grand chose, à quand des logiciels comme Firefox, Safari, Opéra, Thunderbird, Postfix...etc, compatibles DNSSEC.

En ce qui concerne, la firme de Redmond, à ce jour seul Windows 7 possède un résolveur DNSSEC. Et pour la partie serveur, seule la version 2008-R3 peut héberger une zone signée en esclave... à suivre pour ceux qui sont intéressés.

Plus sérieusement, la zone `.fr` devrait être signée courant 2010. Il reste le point `.com` à faire intégralement, à mes yeux c'est une importance capitale et mondiale. En ce qui concerne les racines INTERNET, le mois de mai 2010 devrait annoncer la fin de la migration vers DNSSEC pour toutes les racines.

Je ne peux que conseiller l'utilisation d'OpenDNSSEC pour la gestion de zones avec DNSSEC/NSEC3. C'est réellement un superbe produit très complet et souple. De nombreuses commandes ou de possibilités restent encore à découvrir.

La présence de clés, impose désormais que soit hautement sécuriser les serveurs DNS. Avec DNSSEC/NSEC3 plus rien n'est comme avant en terme DNS.

Annexe A. Les liens utiles.

- Le site OpenDNSSEC : http://www.opendnssec.org
- OpenDNSSEC téléchargement : http://www.opendnssec.org/download/
- OpenDNSSEC WIKI : http://trac.opendnssec.org/wiki
- OpenDNSSEC BugReport : http://trac.opendnssec.org/newticket
- Autre programme OpenDNSSEC (monitor) : http://svn.opendnssec.org/trunk/monitor/
- Le RFC 5155 : http://www.ietf.org/rfc/rfc5155.txt
- Le RFC 4641 : http://www.ietf.org/rfc/rfc4641.txt
- Le RFC 2541 : http://www.ietf.org/rfc/rfc2541.txt
- The DNSSEC Deployment Initiative works to encourage all sectors to voluntarily adopt security measures that will improve security of the Internet's naming infrastructure, as part of a global, cooperative effort that involves many nations and organizations in the public and private sectors. The U.S. Department of Homeland Security Science and Technology Directorate provides support for coordination of the initiative. http://www.dnssec-deployment.org/
- Le site de référence DNSSEC, une mine d'informations et de références (GB): http://www.dnssec.net/
- Un résumé des RFC, des RR et bien d'autres paramètres propres au DNS et DNSSEC (GB): http://www.iana.org/assignments/dns-parameters
- Ce site présente et explique le fonctionnement de DNSSEC. Il présente également plusieurs tableaux pour aider à choisir de bonnes valeurs temporelles (GB) : http://assets.pir.org/PDFs/DNSSEC_Registrar_Webinar_12_02_2009.pdf
- Mini HowTo DNSSEC (FR)http://www.cybervisible.fr/dnssec-dans-la-zone-nl
- Le projet Infrastructure DNSsec et Applications (IDsA)http://www.afnic.fr/afnic/r_d/dnssec
- Wikipdia (FR) pour DNSSEC : http://fr.wikipedia.org/wiki/DNSSEC
- Expos DNSSEC à JRES (Blog de Stéphane Bortzmeyer) (FR) : http://www.bortzmeyer.org/jres-dnssec-2009.html. Il existe plusieurs pages à ce sujet, également sur la sécurisation DNSSEC les racines...à voir.

Les autres projets liés à DNSSEC :

Il faut noter au niveau d'INTERNET et de DNSSEC, la Hollande est particulièrement en avance, et encore mieux est partie prenante, et très active dans DNSSEC, et dans OpenDNSSEC. Ce site explique leurs projets, leurs implications. [http://nlnetlabs.nl/] Il existe en plus d'OpenDNSSEC, d'autres projets qui méritent une certaine attention, comme NSD...

- Drill

  Ce projet est similaire au fameux programme *`Dig`*, mais spécifique pour DNSSEC.

  Drill fournit également une extension pour Firefox, hélas elle n'est pas supportée encore pour les versions récentes (supérieures 3.5.x) ... à suivre.

  http://nlnetlabs.nl/projects/drill/drill_extension.html

- Unbound

  Ce projet est un résolver récursif DNSSEC, avec une option de cache. Vous pouvez trouver plus d'informations sur ce site : http://unbound.net/

- NIST

  National institute Of Standard and Technology. Un excellent document à lire (GB) : DRAFT Secure Domain Name System (DNS) Deployment Guide [http://csrc.nist.gov/publications/PubsDrafts.html#SP-800-81-Rev.%201]

Glossaire

Bind	Bind est le nom d'un logiciel de serveur de noms. C'est un logiciel très connu, et aussi très utilisé partout dans le monde. Le support de DNSSEC est complet, pour les versions récentes de ce logiciel. Voici le site de ISC(Bind) : https://www.isc.org/software/bind
Clé publique	Le système de chiffrement asymétrique nécessite 2 clés différentes. Une clé privée et une clé publique. Comme son nom l'indique, seule la clé publique peut être diffusée sans crainte sur internet. Pour DNSSEC, une zone signée doit comporter au moins 2 clés publiques (1 KSK et 1 ZSK). Cette clé permet de déchiffrer ce que la clé privée a réalisé/chiffrer.
Clé privée	La clé privée est liée à la clé publique dans un système de chiffrement asymétrique. Cette clé sert à chiffrer, elle ne doit pour aucune raison être diffusée.
DNSSEC	(Domain Name System Security Extensions) DNSSEC est un ensemble d'extensions de sécurité pour le protocole DNS.
DS	Correspond à une délégation de signature de la fille vers la mère. Ce type de RR permet d'authentifier la zone fille vis-à-vis de la zone mère.
EDNS	(Extension Mechanism for DNS). C'est une extension de taille des paquets UDP qui peuvent passer de 512 octets à 4096 pour le maximum. Peut poser des problèmes vis-à-vis des pare-feux, il est possible de limiter ceci par `edns-udp-size 1468; max-udp-size 512 ;`. L'activation de ces options doit faire l'objet de plusieurs essais. Elles sont à placer au niveau des options dans le fichier de configuration de Bind. Une obligation pour DNSSEC, ceci se comprend facilement par la taille des données à transmettre. Beaucoup de serveurs (Active directory en particulier) répondent en EDNS, c'est le réflexe d'une réponse à un paquet mal compris par eux. Les réponses EDNS peuvent également arriver suite à une communication sans réponse (time out).
HSM	(Hardware security module). Un HSM est un boîtier hautement sécurisé générateur de clés, qui exporte les clés publiques et protège des clés privées. L'ouverture de ce boîtier doit détruire les clés. Le stockage concerne en particulier les clés privées, aussi les plus sensibles. Vous pouvez aller sur ce site pour de plus amples renseignements à ce sujet : http://fr.wikipedia.org/wiki/Hardware_Security_Module
KSK	Correspond « Key Signing Key ». Cette clé est la plus importante, et la plus sensible pour DNSSEC (toutes les clés sont sensibles). Cette clé (publique) est aussi utilisée comme élément de confiance lors de la l'activation d'une délégation de signature (DS). Cette clé ne signe que le KEY RRset, ou les autres clés. Sa durée de vie est plus longue que la ZSK, mais doit être inférieure à 4 ans (conseillé 1 an). Compte tenu de l'importance de cette clé pour une zone, son encodage est plus fort que la ZSK (conseillé de 2048 à 4096 bits). La signature de cette clé doit être périodiquement renouvelée. Le changement de clé (Rollover) affecte la zone parente, contrairement à la ZSK.
NSEC3	Depuis le mois de mars 2008, NSEC3 est inscrit dans le RFC-5155. C'est aussi la suite de NSEC, et même NSEC v2 dont personne ne parle. NSEC3 est un ensemble de paramètres spécifiques pour DNSSEC. Les RFC de référence ne manquent pas : RFC1034, RFC1035, RFC4033, RFC4034, RFC4035, RFC2136, FC2181, et RFC2308. NSEC3 évite de pouvoir lister une zone en incorporant les RR disponibles pour une zone, en plus des signatures...
OpenDNSSEC	OpenDNSSEC est un projet de gestion de zones qui utilise les paramètres DNSSEC. Ce projet est disponible ici : www.opendnssec.org [http://www.opendnssec.org/]. Le « wiki » et la documentation sont disponibles ici : http://trac.opendnssec.org/wiki. Voici le lien pour poser ou lire les tickets d'incidents/d'erreurs OpenDNSSEC : http://trac.opendnssec.org/report Voici le lien qui permet d'accéder au support (inexistant ce jour ?), aux différentes listes de diffusion : http://www.opendnssec.org/support/
Rollover	Faire rouler, ou permet de faire un roulement de clés (DNSSEC). C'est le remplacement d'une clé par une autre (KSK / ZSK), selon un processus et des paramètres particuliers.

	C'est un procédé relativement long lors d'un changement de clés pour un service DNS DNSSEC.
RRset	Correspond *Ressource Record*. C'est un enregistrement de ressource dans une zone. Exemple : un champ MX dans une zone, correspond un RR car il contient l'adresse IP, un TTL, une classe ...
RRSIG	Lié à DNSsec, il représente la signature (DNSSEC) pour une donnée de ressources. Plusieurs types MX (exemple) donnent seulement une seule signature. Le RFC de référence est ici : RFC 4034 [http://rfc-ref.org/RFC-TEXTS/4034/chapter3.html]
TLD	Le « Top Level Domain » représente les serveurs au plus haut niveau dans la hiérarchie DNS. Donc c'est les 13 ROOT déclarés (ou plus). Le terme de TLDN est aussi un équivalent, la lettre « N », veut dire « name ». Et en dessous des TLD, il existe une multitude de sous-classes comme : « ccTLD (country code top-level domain) » qui représente 2 lettres pour identifier les divers pays, dont le ".fr". « gTLD (generic top-level domain) » qui gère les particules ".net", ".org"...etc Et encore bien d'autres encore, la liste pourrait être longue...
ZSK	Correspond à « Zone Signing Key ». Cette clé sert à signer une zone complète. La signature de cette clé est réalisée par la KSK; comme tout enregistrement KEY d'une zone. Comparé à la KSK, son encodage est moindre (conseillé 1024 bits). En conséquence, la signature de cette clé doit être périodiquement renouvelée. La clé privée est alors employée pour signer numériquement tous les enregistrements, et ce en incluant l'enregistrement de la clé publique. Ces signatures sont placées dans les enregistrements SIG. La mise à jour d'une clé ZSK n'affecte que la zone concernée, contrairement à la KSK.