Superviser ses signatures DNSSEC

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 7 avril 2014. Dernière mise à jour le 15 mars 2019

https://www.bortzmeyer.org/monitor-dnssec.html

Le système DNSSEC permet d'authentifier les données distribuées via le DNS et protège donc ainsi des attaques par empoisonnement. C'est un outil nécessaire dans la boîte à outils de sécurisation du DNS. Mais rien n'est gratuit en ce bas monde : la cryptographie protège mais elle complique les choses et crée des risques. Parmi ceux-ci, le risque d'expiration des signatures. Il est donc nécessaire de superviser ses signatures DNSSEC. Comment ? Et avec quoi ?

Si vous voulez une introduction à DNSSEC en français, vous pouvez lire mon exposé à JRES <https://www.bortzmeyer.org/jres-dnssec-2009.html>. Notez-y un point important : les signatures des enregistrements DNSSEC **expirent** au bout d'un moment et il est donc nécessaire de re-signer de temps en temps. Si ce processus de re-signature ne marche pas, les signatures vont finir par expirer, rendant le domaine inutilisable. Voici un exemple de signature DNSSEC :

```
% dig +dnssec A www.bortzmeyer.org
...
;; ANSWER SECTION:
www.bortzmeyer.org. 68585 IN A 204.62.14.153
www.bortzmeyer.org. 68585 IN RRSIG A 8 3 86400 20140414143956 (
20140325120748 15774 bortzmeyer.org.
dgJ3BOjUz3hdlRWEbcFK14Jqyl+az/tas/dKEcBs/2QK
4vUd2VihXtEmpLQ6D+FVtMh6n7OubrEpLezEGkHtXKOe
3FO6l+EhrjH82BwjGGnd50RMNDHGk8IR0TOsOj/cNGZM
V4Gj24mOV5ANbYWxlqWXYPl9BVi81MVhluw9sas= )
...
```

On voit dans la réponse l'adresse IPv4 du serveur et la signature (enregistrement RRSIG). dig formate la signature de manière lisible par un humain, avec une date de mise en service ("*inception*") au 25 mars et une date d'**expiration** au 14 avril. Ce jour-là, à 14h39 UTC, la signature expirera (le but est d'éviter les attaques par rejeu). Il faudra donc, avant, signer à nouveau. Dans le cas de bortzmeyer.org, c'est fait avec OpenDNSSEC. On pourrait aussi lancer un dnssec-signzone ou bien un ldns-signzone depuis cron. Ou laisser BIND prendre cela en charge avec son mécanisme de signature automatique.

Mais toutes ces solutions ont un point commun, elles sont fragiles. Pendant des mois, elles fonctionnent puis, un jour, un léger changement fait qu'elles ne marchent plus et qu'on risque de ne pas s'en apercevoir. Un exemple qui m'était arrivé en changeant de version de Debian : la bibliothèque de SoftHSM <http://www.opendnssec.org/softhsm/> avait changé d'emplacement <https://twitter.com/bortzmeyer/status/347458342639247360>, le fichier de configuration d'OpenDNSSEC pointait donc au mauvais endroit et le signeur d'OpenDNSSEC ne tournait donc plus. Quelques temps après, bortzmeyer.fr expirait <https://twitter.com/bortzmeyer/status/347438625203560448>...

Le problème n'est pas spécifique à DNSSEC. Dans toutes les solutions de sécurité, il y a des dates limites, conçues pour éviter qu'un méchant qui aurait mis la main sur des informations secrètes puisse les utiliser éternellement. C'est par exemple pour cela que les certificats X.509 ont une date d'expiration (attention, avec DNSSEC, les clés n'expirent pas, seules les signatures le font). Comme le savent les utilisateurs de HTTPS, il est très fréquent que le webmestre oublie de renouveler les certificats et paf. À part des bonnes procédures (rappel mis dans l'agenda...), la solution est de superviser. Tout responsable sérieux d'un site Web HTTPS supervise l'expiration <https://www.bortzmeyer.org/tester-expiration-certifs.html>. Il faut faire la même chose pour DNSSEC.

La solution que je vais montrer ici fonctionne avec ma configuration Icinga <https://www.bortzmeyer.org/icinga.html> mais elle ne repose que sur des outils compatibles avec l'API Nagios donc elle devrait marcher avec beaucoup d'outils de supervision.

Après plusieurs essais (voir les notes de ces essais plus loin), j'ai choisi comme outil de base le script de test de Duane Wessels <http://dns.measurement-factory.com/tools/nagios-plugins/check_zone_rrsig_expiration.html>. Ses spécifications collent parfaitement à ce que je veux : il se connecte à tous les serveurs DNS d'une zone, demande les signatures, regarde les dates d'expiration et peut signaler un avertissement ou une erreur selon des seuils choisis par l'utilisateur. Un exemple à la main :

```
% perl check_zone_rrsig_expiration -Z nic.fr
ZONE OK: No RRSIGs expiring in the next 3 days; (1.04s) |time=1.042905s;;;0.000000
```

On peut choisir les seuils, donc mettons qu'on veut un avertissement s'il reste moins d'une semaine :

```
% perl check_zone_rrsig_expiration -Z nic.fr -W 7
ZONE WARNING: MX RRSIG expires in 3.7 days at ns6.ext.nic.fr; (0.28s) |time=0.281515s;;;0.000000
```

Pour installer et exécuter ce script, il faut Perl et certains modules indiqués dans la documentation. Sur ma machine Arch Linux, ils n'étaient pas en paquetage standard, il faut donc utiliser AUR, un dépôt non officiel, accessible avec les commandes pacaur ou yaourt :

```
% yaourt  -S perl-net-dns perl-net-dns-sec
```

Attention, si vous n'installez pas tous les paquetages indiqués dans la documentation, vous aurez un message pas clair du tout :

```
***  WARNING!!!  The program has attempted to call the method
***  "sigexpiration" for the following RR object:
```

Une fois le programme correctement installé, je vous recommande l'option `-d` si vous voulez déboguer en détail ce qu'il fait.

On configure ensuite Icinga, par exemple :

```
  object Host NodeName {
  import "generic-host"
  address = "127.0.0.1"
  address6 = "::1"
  vars.role = "AuthDNS"
  [...]
}

object CheckCommand "CheckDNSSEC" {
  import "plugin-check-command"
  command = [ PluginDir + "/check_zone_rrsig_expiration" ]
  arguments = {
    "-Z" = "$zone$"
    "-W" = "$warn$"
    "-C" = "$crit$"
  }
  vars.warn = "14"
  vars.crit = "7"
}

apply Service "DNSSEC" {
  check_command = "CheckDNSSEC"

  check_interval = 86400
  vars.zone = "example.org"
  assign where host.vars.role == "AuthDNS"
}
```

(Merci à Xavier Humbert à ce sujet.) Cette configuration était pour Icinga 2. Pour Icinga 1 :

```
define command {
        command_name    check-zone-rrsig
        command_line    /usr/local/sbin/check_zone_rrsig_expiration -Z $HOSTADDRESS$ -W $ARG1$ -C $ARG2$
        }

...

define service {
       use dns-rrsig-service
       hostgroup_name My-zones
       service_description SIGEXPIRATION
       # Five days left: warning. Two days left: panic.
       check_command   check-zone-rrsig!5!2
}

define host{
        name                            my-zone
        use                     generic-host
check_command             check-always-up
        check_period                    24x7
        check_interval                  5
        retry_interval                  1
        max_check_attempts              3
        contact_groups                  admins
        notification_period             24x7
        notification_options u,d,r
        register 0
}
```

```
define hostgroup{
       hostgroup_name My-zones
       members bortzmeyer.org,bortzmeyer.fr,etc-etc
}

define host{
       use moi-zone
       host_name bortzmeyer.fr
}
```

Une fois que c'est fait, on redémarre Icinga. Ici, voici un test avec une zone délibérement cassée (elle a été signée manuellement en indiquant la date d'expiration `ldns-signzone -e 20140322100000 -o broken.rd.nic.fr. broken.rd.nic.fr Kbroken.rd.nic.fr.+008+15802` et sans re-signer ensuite). Icinga enverra ce genre d'avertissement :

```
Notification Type: PROBLEM

Service: DNSRRSIG
Host: broken.rd.nic.fr
Address: broken.rd.nic.fr
State: WARNING

Date/Time: Fri Mar 28 06:50:38 CET 2014

Additional Info:

ZONE WARNING: DNSKEY RRSIG expires in 1.2 days at ns2.bortzmeyer.org: (1.12s)
```

Puis un `CRITICAL` puis, lorsque la zone aura vraiment expiré :

```
Notification Type: PROBLEM

Service: DNSRRSIG
Host: broken.rd.nic.fr
Address: broken.rd.nic.fr
State: CRITICAL

Date/Time: Mon Mar 31 09:10:38 CEST 2014

Additional Info:

ZONE CRITICAL: ns2.bortzmeyer.org has expired RRSIGs: (1.10s)
```

Si on re-signe à ce moment, le problème disparait :

```
Notification Type: RECOVERY

Service: DNSRRSIG
Host: broken.rd.nic.fr
Address: broken.rd.nic.fr
State: OK

Date/Time: Mon Mar 31 09:40:38 CEST 2014

Additional Info:

ZONE OK: No RRSIGs expiring in the next 3 days: (1.04s)
```

Et, dans le journal d'Icinga, cela apparaitra :

```
[Fri Mar 21 21:40:32 2014] SERVICE ALERT: broken.rd.nic.fr;DNSRRSIG;CRITICAL;SOFT;2;ZONE CRITICAL: DNSKEY RRSIG
...
[Fri Mar 21 21:42:32 2014] SERVICE ALERT: broken.rd.nic.fr;DNSRRSIG;OK;SOFT;3;ZONE OK: No RRSIGs expiring in the
```

Pour les utilisateurs d'OpenDNSSEC, le paramètre important à régler en même temps que les seuils de la supervision est le paramètre `<Refresh>`. Comme le dit la documentation `<https://wiki.opendnssec.org/display/DOCS/kasp.xml#kasp.xml-Signatures>` : « *"The signature will be refreshed when the time until the signature expiration is closer than the refresh interval."* » Donc, en pratique, c'est la durée qu'il faut indiquer avec l'option `-C` (seuil critique). Attention, OpenDNSSEC ajoute de légères variations ("*jitter*").

J'ai indiqué plus haut qu'il y avait des alternatives à la solution finalement choisie. Il en existe même une liste sur le site d'Icinga `<https://wiki.icinga.org/display/howtos/DNS+Monitoring>`. Voici quelques pistes avec mes commentaires.

J'aurais pu développer une solution complète avec Python et dnspython `<https://www.bortzmeyer.org/dnspython.html>` qui a une bonne gestion de DNSSEC. J'ai réalisé un prototype mais, dans la vie, il faut savoir reconnaître un logiciel meilleur que le sien.

Il y a un outil développé par les gens de `.se`, dnssec_monitor `<https://github.com/dotse/dnssec-monitor#readme>`. Écrit en Perl, il a les mêmes dépendances que le script choisi. Mais je n'arrive pas réellement à comprendre comment il s'intègre avec Nagios.

Il existe un outil en Ruby dans OpenDNSSEC `<https://github.com/opendnssec/dnssec-monitor/blob/master/README>` (il est même décrit en détail dans la documentation d'Icinga `<https://wiki.icinga.org/display/howtos/check_dnssec>`). Il a pas mal de dépendences Ruby donc j'ai renoncé.

L'outil nagval `<https://github.com/jpmens/nagval>` est très bien mais il n'a pas le même cahier des charges et, notamment, il ne permet pas de tester l'expiration qui va survenir.

Il existe un énorme ensemble de programmes de tests Nagios `<https://git.durchmesser.ch/monitoringplug/monitoringplug>` qui semble intéressant, et qui compte un `check_dnssec_expiration`. Il se compile bien :

```
% ./configure --without-man
% make
```

mais l'exécution est incohérente, une fois sur deux :

```
% ./dns/check_dnssec_expiration -v -D nic.fr -w 20 -c 3
CRITICAL - SOA is not signed.
```

La bogue a été signalée à l'auteur mais pas encore résolue. Il semble que l'outil soit très sensible au résolveur utilisé et qu'il faille forcer (via `resolv.conf` ou via l'option `-H`) un résolveur rapide et fiable.

Mat a également un script à lui <https://gist.github.com/mat813/8114791#file-check-expire-awk en awk (avec une version en shell <https://gist.github.com/mat813/8114791#file-check-expire>). Mais il est à mon avis très dépendant d'un environnement local, pas utilisable tel quel, sans sérieuses modifications, à mon avis. Je cite l'auteur : « c'est tout à fait adaptable, il suffit de remplacer `/usr/bin/make -VSIGNED` par la liste des fichiers de zones signés et `/usr/bin/make -VSIGNED:R:T` par l'ensemble des noms des zones. `SIGNED` étant défini dans le `Makefile` comme `SIGNED!= find -s * -name '*.signed'`. ». Du même auteur, un outil <https://gist.github.com/mat813/8114791#file-check-exp permet de tester des fichiers, pas facilement des zones vivantes :

```
% cat *.signed | awk -f check-expire.awk
```

Pour tester une zone vivante, il faut que le transfert de zone soit autorisé :

```
% dig +noall +answer axfr @$SERVERNAME $ZONE | awk -f check-expire.awk
```

Je n'ai pas encore testé l'outil check_dnssec_expiry <https://github.com/mrimann/check_dnssec_expiry>.

Enfin, SURFnet a développé un outil <https://dnssec.surfnet.nl/?p=562> (qui dépend de ldns <http://www.nlnetlabs.nl/projects/ldns/>) mais que je n'ai pas réussi à compiler :

```
...
cc -lcrypto `ldns-config --libs` -o sigvalcheck sigvalcheck.o rrsig_valcheck.o query.o
rrsig_valcheck.o: In function `ldns_rrsig_time_until_expire':
/home/stephane/tmp/sigvalcheck-0.1/rrsig_valcheck.c:42: undefined reference to `ldns_rr_rrsig_expiration'
/home/stephane/tmp/sigvalcheck-0.1/rrsig_valcheck.c:41: undefined reference to `ldns_rdf2native_time_t'
```