

Premiers essais avec Cascade, le logiciel pour gérer ses zones DNSSEC

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 13 octobre 2025

<https://www.bortzmeyer.org/cascade-debut.html>

Annoncé officiellement le 7 octobre <<https://social.nlnetlabs.nl/@nlnetlabs/115331567449071853>>, Cascade <<https://blog.nlnetlabs.nl/cascade/>> est le successeur d'OpenDNSSEC. Ce programme sert à gérer automatiquement les opérations répétitives liées à DNSSEC comme la re-signature ou le remplacement d'une clé. (Notez que Cascade évolue vite en ce moment et que cet article ne sera **pas** mis à jour. J'en écrirai d'autres <<https://www.bortzmeyer.org/cascade-deux.html>>.)

OpenDNSSEC <<https://www.bortzmeyer.org/opendnssec-debut.html>> est, depuis 15 ans, le logiciel utilisé par mes domaines personnels (comme `bortzmeyer.org` par lequel vous êtes passé pour voir cet article) mais aussi par plusieurs TLD comme `.fr`. Mais il est désormais en fin de vie (cela a été annoncé le 3 octobre <<https://lists.opendnssec.org/pipermail/opendnssec-announce/2025-October/000150.html>>), il n'y a plus que les mises à jour de sécurité. La même organisation, NLnet Labs <<https://nlnetlabs.nl/>>, a développé un successeur, Cascade <<https://blog.nlnetlabs.nl/cascade/>>. Celui-ci est actuellement en alpha (**ne l'utilisez pas pour la production!**) et voici un premier essai.

DNSSEC, contrairement à ce qui se passe avec le DNS d'antan, nécessite des actions périodiques. La plus évidente est la re-signature, car les signatures DNSSEC ont une date d'expiration (pour éviter les attaques par rejeu). Ici, cette signature expire le 18 octobre 2025 (le « 20251018235702 ») :

```
% dig +dnssec cyberstructure.fr TXT
...
;; ANSWER SECTION:
cyberstructure.fr. 86400 IN TXT "v=spf1 mx -all"
cyberstructure.fr. 86400 IN RRSIG TXT 13 2 86400 (
  20251018235702 20251004235301 34065 cyberstructure.fr.
  Hg/ModpW3pTLgUicgp7yrVUXyVAXdEIOTfONpN3pxG0
  ...
```

Mais il y a aussi la nécessité de pouvoir changer les clés cryptographiques utilisées, par exemple en cas de compromission, avérée ou suspectée. (La plupart des organisations changent les clés systématiquement, compromission ou pas, notamment pour être sûr de pouvoir le faire sans incident le jour où ça sera vraiment indispensable.) Cascade, comme OpenDNSSEC avant lui, automatise toutes ces tâches.

Cascade est développé en Rust, le langage à la mode, et qui pourrait remplacer une bonne partie du logiciel d'infrastructure de l'Internet, qui est quasi-uniquement en C (et C++). Il doit son nom au fait qu'une zone DNS qu'on va signer passe par plusieurs étapes successives, où on peut insérer différentes opérations, par exemple de validation.

Jouons donc un peu avec Cascade (j'ai bien dit que c'était en **version alpha** ; ne le faites pas en production). Notez que vous pouvez lire la documentation officielle <<https://cascade.docs.nlnetlabs.nl/en/latest/quick-start.html>> au lieu de cet article. Mais j'ai bien dit que le logiciel était en version alpha, ne vous étonnez pas s'il manque beaucoup de choses dans la documentation (et dans le code <<https://cascade.docs.nlnetlabs.nl/en/latest/limitations.html>>, bien sûr). Il existe déjà des paquetages tout faits (mais évidemment pas encore distribués avec votre système d'exploitation) mais on va compiler, c'est plus amusant. Il faut donc une version de Rust qui marche. Rust est très pénible, avec ses changements permanents, qui font que certain-es conseillent même de récupérer et d'installer automatiquement une nouvelle version du compilateur chaque nuit. Mais, bon, la version qui est dans Debian "*stable*" (la version 13 de Debian) marche bien. Cascade est un logiciel libre, et il est hébergé <<https://github.com/NLnetLabs/cascade>> sur GitHub :

```
% git clone https://github.com/NLnetLabs/cascade.git
...

# Installer les dépendances
% apt install cargo pkg-config libssl-dev
...

% rustc --version
rustc 1.85.0 (4d91de4e4 2025-02-17) (built from a source tarball)

% cd cascade

% cargo build
...
```

Et vous n'avez plus qu'attendre que votre machine compile quelques centaines de "*crates*" Rust. Vous vous retrouvez alors avec deux exécutables, `./target/debug/cascade` et `./target/debug/cascaded`. Le second est le démon qui va tourner en permanence. (Si vous venez d'OpenDNSSEC, notez qu'il n'y a cette fois qu'un seul démon.) Notez que ce démon, bien qu'il inclue un serveur DNS minimal, ne doit **pas** être exposé sur l'Internet. La configuration recommandée est de l'utiliser comme serveur maître caché, sur lequel s'alimenteront un ou plusieurs des serveurs faisant autorité <<https://www.bortzmeyer.org/serveur-dns-faisant-autorite.html>> pour vos domaines. (Si vous venez d'OpenDNSSEC, notez que Cascade ne sait actuellement pas écrire la zone sur le disque, il faut faire un transfert de zones, cf. RFC 5936¹.) Le premier exécutable, lui, est le programme que vous lancez pour accomplir telle ou telle tâche, et qui parle au démon, qui fera le boulot.

Sinon, une autre solution, pour compiler, est de suivre les instructions <<https://cascade.docs.nlnetlabs.nl/en/latest/installation.html>> :

1. Pour voir le RFC de numéro NNN, <https://www.ietf.org/rfc/rfcNNN.txt>, par exemple <https://www.ietf.org/rfc/rfc5936.txt>

```
cargo install --locked --git https://github.com/nlnetlabs/cascade
cargo install --locked --branch keyset --git https://github.com/nlnetlabs/dnst
```

Et les exécutables atterriront dans votre `/.cargo/bin`.

Regardons les options des deux programmes :

```
% ./target/debug/cascade
Usage: cascade [OPTIONS] <COMMAND>
```

```
Commands:
  config    Manage Cascade's configuration
  zone      Manage zones
  policy    Manage policies
  keyset    Execute manual key roll or key removal commands
  hsm       Manage HSMs
  template  Print example config or policy files
```

```
Options:
  -s, --server <IP:PORT>  The cascade server instance to connect to [default: 127.0.0.1:4539]
  --log-level <LEVEL>    The minimum severity of messages to log [default: warning] [possible values: trace, debug, info, warning, error, critical]
  -h, --help              Print help
  -V, --version           Print version
```

```
% ./target/debug/cascaded -h
Usage: cascaded [OPTIONS]
```

```
Options:
  --check-config          Check the configuration and exit
  --state <PATH>         The global state file to use
  -c, --config <PATH>    The configuration file to load
  --log-level <LEVEL>    The minimum severity of messages to log [possible values: trace, debug, info, warning, error, critical]
  -l, --log <TARGET>     Where logs should be written to [possible values: stdout, stderr, file:<PATH>, syslog]
  -d, --daemonize        Whether Cascade should fork on startup
  -h, --help             Print help
  -V, --version          Print version
```

Si vous avez procédé comme moi, un dernier truc à compiler, il nous faut aussi `cascade-dnst`, par la même organisation :

```
% cargo install --locked --branch keyset --git https://github.com/nlnetlabs/dnst
```

Vérifiez bien que les exécutables sont dans votre PATH (par exemple, que `/.cargo/bin` y soit). Pour le démon, vous pouvez aussi utiliser la variable `dnst-binary-path` dans le `config.toml`.

<https://www.bortzmeyer.org/cascade-debut.html>

Pour utiliser Cascade, vous devrez, comme avec OpenDNSSEC, définir une politique (quel algorithme de cryptographie, quelle durée de vie des signatures, quand remplacer les clés, etc), indiquer quelles zones Cascade doit gérer, et avoir un serveur faisant autorité qui va récupérer la zone sur Cascade.

Maintenant, configurons le logiciel, toujours en suivant la documentation <<https://cascade.docs.nl.netlabs.nl/en/latest/quick-start.html>>. (Si vous oubliez cette étape, vous aurez un message « *"Cascade couldn't be configured : could not load the config file '/etc/cascade/config.toml' : No such file or directory (os error 2)"* ».) Vous avez avec Cascade un fichier de configuration d'exemple, `./etc/config.template.toml`, à la syntaxe TOML. Le fichier d'exemple peut être utilisé tel quel dans beaucoup de cas. J'ai juste modifié les paramètres de journalisation :

```
[daemon]
log-level = "debug"
log-target = { type = "file", path = "/dev/stdout" }
# Mais log-target = { type = "syslog" } est bien aussi.
```

Lançons maintenant le démon (voyez plus loin les mécanismes pour l'avoir en permanence). D'abord, on crée les répertoires configurés dans le `config.toml` :

```
% sudo mkdir /var/lib/cascade
% sudo chown $USER /var/lib/cascade

% sudo mkdir /etc/cascade/policies
```

Puis on y va :

```
% ./target/debug/cascaded
[2025-10-09T15:56:07.480Z] INFO cascaded: State file not found; starting from scratch
...
[2025-10-09T15:56:09.182Z] DEBUG cascade::state: Saved the global state (to '/var/lib/cascade/state.db')
...
```

Maintenant que le démon tourne, testons le programme de contrôle :

```
% ./target/debug/cascade zone list
%
```

Pas de message d'erreur, mais pas de zones configurées, c'est normal. (Si le démon ne tournait pas, vous auriez eu « *"[2025-10-09T15:59:37.183Z] ERROR cascade : Error : HTTP request failed : error sending request for url (http://127.0.0.1:4539/zone/)"* ».)

Maintenant, il faut créer une politique. La plus courte est :

```
version = "v1"
```