RFC 8633 : Network Time Protocol Best Current Practices

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 27 octobre 2019

Date de publication du RFC : Juillet 2019

https://www.bortzmeyer.org/8633.html

Le protocole NTP, qui sert à synchroniser les horloges sur l'Internet, est probablement un des plus vieux protocoles encore en service. Il est normalisé dans le RFC 5905 [1]. Ce nouveau RFC ne change pas la norme, il rassemble simplement un ensemble de bonnes pratiques pour l'utilisation de NTP. Sa lecture est donc très recommandée à toutes les personnes qui gèrent la synchronisation d'horloges dans leur organisation.

Bien sûr, tout dépend des niveaux d'exigence de cette organisation. Quel écart entre les horloges acceptez-vous ? Quel est le risque d'une attaque délibérée contre les serveurs de temps que vous utilisez ? À vous de lire ces bonnes pratiques et de les adapter à votre cas.

Le RFC commence par un conseil de sécurité réseau général (section 2). NTP est fondé sur UDP, ce qui signifie que le protocole de transport ne protège pas contre les usurpations d'adresses <https://www.bortzmeyer.org/usurpation-adresse-ip.html>. Et comme les réponses NTP sont souvent bien plus grosses (en octets) que les questions, des attaques par réflexion et amplification <https://www.bortzmeyer.org/ntp-reflexion.html> sont possibles, et effectivement observées dans la nature <https://www.bortzmeyer.org/attaque-ntp-en-vrai.html>. (Le RFC recommande également la lecture de « "*Technical Details Behind a 400Gbps NTP Amplification DDoS Attack*" <https://blog.cloudflare.com/technical-details-behind-a-400gbps-ntp-amplification-ddos-attack> », de « "*Taming the 800 Pound Gorilla : The Rise and Decline of NTP DDoS Attacks*" <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.680.9607> » et de « "*Attacking the Network Time Protocol*" <https://eprint.iacr.org/2015/1020.pdf> ».) Il est donc nécessaire que les opérateurs réseau déploient les mesures « BCP 38 <https://www.bortzmeyer.org/bcp38.html> » contre les usurpations d'adresses IP.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5905.txt

Après ce conseil général, qui concerne également d'autres protocoles, comme DNS ou SNMP, la section 3 du RFC se penche sur les bonnes pratiques de configuration de NTP. Évidemment, il faut maintenir les logiciels à jour (ce qu'on oublie plus facilement lorsqu'il s'agit d'un protocole d'infrastructure, comme NTP).

Moins évidente, la nécessité d'avoir plusieurs sources de temps. Une source donnée peut être malveillante, ou tout simplement incorrecte. Or, NTP peut utiliser plusieurs sources, et trouver le temps correct à partir de ces sources (les détails sont dans le RFC 5905, et dans le livre de D. Mills, « "*Computer network time synchronization : the Network Time Protocol*" ».) Trois sources sont le minimum, quatre sont recommandées, si, bien sûr, elles sont suffisamment diverses, et dignes de confiance. (Au passage, Renater gère une liste de serveurs NTP en France <https://services.renater.fr/ntp/serveurs_francais> mais elle ne semble pas à jour, chronos.cru.fr n'existe plus, il y manque le serveur de l'AFNIC, ntp.nic.fr, etc.)

Ce conseil de chercher plusieurs serveurs suppose évidemment que ces serveurs sont indépendants : s'ils prennent tous le temps depuis une même source et que celle-ci est déréglée ou malveillante, avoir plusieurs serveurs ne suffira pas. Il est donc également nécessaire de veiller à la diversité des horloges sous-jacentes. Avoir plusieurs serveurs mais connectés à des horloges du même vendeur fait courir le risque d'un problème commun à toutes. La diversité doit aussi s'appliquer au choix du type d'horloge : si on utilise plusieurs horloges, mais toutes fondées sur des constellations de satellites, une tache solaire va les perturber tous en même temps. Autre risque : un problème DNS supprimant le nom de domaine, comme c'était arrivé à usno.navy.mil (l'USNO), en décembre 2018 et surtout à ntp.org en janvier 2017 (cf. cette discussion <https://gist.github.com/bortzmeyer/8643239b33556257750a10a2cfd46600 ou bien celle-ci <https://community.ntppool.org/t/dns-resolving-problems-for-ch-pool-ntp-or 136>).

Autre question, les messages de contrôle de NTP. Introduits dans l'annexe B du RFC 1305, qui normalisait la version 3 de NTP, ils n'ont pas été conservés pour la version 4 (RFC 5905). (Un projet existe à l'IETF pour les remettre, cf. draft-ietf-ntp-mode-6-cmds.) Utiles à l'administrateur système pour la gestion de ses serveurs, ces messages peuvent être dangereux, notamment en permettant des attaques par réflexion, avec amplification <https://www.bortzmeyer.org/ntp-reflexion.html>. La bonne pratique est donc de ne pas les ouvrir au monde extérieur, seulement à son réseau. Des exemples de configurations restrictives figurent à la fin de cet article.

Il est évidemment nécessaire de superviser ses serveurs NTP, afin de s'assurer qu'ils sont en marche, et, surtout, du fait qu'ils soient bien synchronisés. Des exemples, utilisant Icinga, figurent à la fin de cet article.

Un serveur NTP qui sert des dizaines de milliers de clients peut nécessiter beaucoup de ressources réseau. Il est donc important de n'utiliser comme serveur que des serveurs qu'on est autorisé à questionner (ce qui est le cas des serveurs publics comme ntp.nic.fr). Il existe hélas de nombreux exemples d'abus de serveurs NTP, le plus célèbre étant sans doute celui du serveur de Poul-Henning Kamp par D-Link.

Pour permettre à tous et toutes de synchroniser leurs horloges, le projet « "*NTP Pool*" <http://www.pool.ntp.org/> » a été créé. De nombreux volontaires mettent à la disposition de tous leurs serveurs NTP. L'heure ainsi distribuée est en général de bonne qualité mais, évidemment, le projet ne peut fournir aucune garantie. Il convient bien pour les configurations par défaut distribuées avec les logiciels, ou pour des machines non critiques. Autrement, il faut utiliser des serveurs « de confiance ».

Pour l'utiliser, il faut regarder les instructions <https://www.pool.ntp.org/en/use.html> (elles existent aussi en français <https://www.pool.ntp.org/fr/use.html>). En gros, on doit indiquer comme serveurs NTP des noms pris sous pool.ntp.org et ces noms pointeront, au hasard, vers des machines différentes, de manière à répartir la charge. Voici un exemple (avec un serveur français <https://www.pool.ntp.org/zone/fr>) :

```
% dig +short A 0.fr.pool.ntp.org
5.196.192.58
51.15.191.239
92.222.82.98
162.159.200.123
```

Mais quelque temps après, les adresses IP auront changé.

```
%  dig +short A 0.fr.pool.ntp.org
80.74.64.2
212.83.154.33
94.23.99.153
37.187.5.167
```

Voici un exemple de configuration avec le serveur NTP habituel <https://support.ntp.org/bin/view/Support/ConfiguringNTP>, dans son `ntp.conf` :

```
pool 0.fr.pool.ntp.org iburst
pool 1.fr.pool.ntp.org iburst
pool 2.fr.pool.ntp.org iburst
pool 3.fr.pool.ntp.org iburst
```

Ainsi, on aura quatre serveurs, pointant vers des adresses réparties dans le lot. Avec OpenNTPd <http://www.openntpd.org/>, ce serait :

```
servers fr.pool.ntp.org
```

L'administrateur système ne pense pas toujours à ajuster la configuration, donc beaucoup de fournisseurs de logiciels ont un sous-domaine <https://www.pool.ntp.org/en/vendors.html> de `pool.ntp.org`, utilisé dans les configurations livrées avec leurs serveurs NTP. Par exemple, pour OpenWrt, le fichier de configuration par défaut contiendra :

```
server 0.openwrt.pool.ntp.org iburst
server 1.openwrt.pool.ntp.org iburst
server 2.openwrt.pool.ntp.org iburst
server 3.openwrt.pool.ntp.org iburst
```

Un problème récurrent des horloges sur l'Internet est celui des secondes intercalaires. La Terre étant imparfaite, il faut de temps en temps corriger le temps universel avec ces secondes intercalaires. Jusqu'à présent, on a toujours ajouté des secondes, puisque la Terre ralentit, mais, en théorie, on pourrait avoir à en retirer. Il existe un temps qui n'a pas ce problème, TAI, mais, outre qu'il s'éloigne petit à petit du temps astronomique, il n'a pas été retenu dans les normes comme POSIX (ou NTP, qui ne connait qu'UTC…) Il faut donc gérer les soubresauts d'UTC, et c'est une source de bogues sans fin. Les secondes intercalaires ne sont pas prévisibles longtemps à l'avance (je vous avait dit que la Terre est imparfaite) et il faut donc lire les bulletins de l'IERS <https://www.iers.org/IERS/EN/

Publications/Bulletins/bulletins.html> (en l'occurrence le bulletin C) pour se tenir au courant. Notez que ce bulletin n'est pas écrit sous une forme structurée, lisible par un programme, donc on pourra préférer le leap-seconds.list, disponible en plusieurs endroits. Pour un serveur NTP, une autre solution est d'utiliser des horloges qui distribuent de l'information sur les secondes intercalaires prévues. C'est le cas du GPS ou de DCF77. Dans ce cas, le serveur NTP peut se préparer un peu à l'avance (ce qui n'évite pas les bogues...)

Autre problème amusant, noté par le RFC, le "*leap smearing*", qui consiste à lisser l'arrivée d'une seconde intercalaire au lieu de brutalement décaler l'horloge d'une seconde. Lors de la seconde intercalaire de juin 2015, certains serveurs NTP faisaient du "*leap smearing*" et pas d'autres, ce qui semait la confusion chez les clients qui avaient un mélange de ces deux types de serveurs. Le "*leap smearing*" n'est pas conforme à la norme NTP et ne doit donc **pas** être utilisé sur des serveurs NTP publics, d'autant plus que le protocole ne permet pas de savoir si le serveur utilise ce "*smearing*" ou pas. Dans un environnement fermé, par contre, on fait évidemment ce qu'on veut.

Concernant ce "*leap smearing*", le RFC note qu'il peut poser des problèmes juridiques : l'horloge de la machine ne sera pas en accord avec l'heure légale, ce qui peut créer des histoires en cas, par exemple, d'accès des autorités aux journaux.

Passons maintenant aux mécanismes de sécurité de NTP (section 4 du RFC). L'analyse des risques a été faite dans le RFC 7384, notre RFC rappelle les moyens de faire face à ces risques. Il y a bien sûr les clés partagées (la directive keys /etc/ntp/ntp.keys dans le serveur NTP classique). NTP n'a pas de mécanisme de distribution de ces clés, il faut le faire en dehors de NTP (copier /etc/ntp/ntp.keys...), ce qui ne marche donc pas pour des serveurs publics. (Comme il y a toujours des lecteurs qui me disent « mais c'est pas pratique de recopier les clés à la main sur toutes les machines », je rappelle l'existence d'Ansible, et autres outils analogues.) À noter que le seul algorithme de condensation normalisé pour l'utilisation de ces clés est MD5, clairement dangereux (RFC 6151). Ceci dit, d'autres algorithmes sont parfois acceptés par les mises en œuvre de NTP, cf. RFC 8573. (Opinion personnelle : MD5 vaut mieux que pas de sécurité du tout.)

Et...c'est tout. Il n'existe pas actuellement d'autre mécanisme de sécurité pour NTP. Le système Autokey, normalisé dans le RFC 5906 a été abandonné, en raison de ses vulnérabilités <https://lists.ntp.org/pipermail/ntpwg/2011-August/001714.html>. Un travail était en cours à l'époque pour lui concevoir un successeur, ce qui a donné le RFC 8915.

La section 5 de notre RFC résume les bonnes pratiques en matière de sécurité NTP :

— Éviter les fuites d'information que permettent les requêtes de contrôle. Les articles de Malhotra, A., Cohen, I., Brakke, E., et S. Goldberg, « "*Attacking the Network Time Protocol*" <https://eprint.iacr.org/2015/1020.pdf> », de Van Gundy, M. et J. Gardner, « "*Network Time Protocol Origin Timestamp Check Impersonation Vulnerability*" <https://www.talosintel.com/reports/TALOS-2016-0077> » (CVE-2015-8138) et de Gardner, J. et M. Lichvar, « "*Xleave Pivot : NTP Basic Mode to Interleaved*" <https://blog.talosintel.com/2016/04/vulnerability-spotl
27.html> » (CVE-2016-1548) documentent des attaques rendues possibles par ce côté trop bavard de NTP. C'est ainsi par exemple que Shodan se permettait de repérer <https://arstechnica.com/information-technology/2016/02/using-ipv6-with-linux-youve-likely-been-visite
> des adresses IPv6 à analyser, pour compenser le fait que l'espace d'adressage IPv6 est trop gros pour le balayer systématiquement (RFC 7707).
— Surveiller les attaques connues, par exemple avec Suricata.
— NTP a un mécanisme de répression des requêtes, le KoD ("*Kiss-o'-Death*", RFC 5905, section 7.4), qui permet de calmer les clients qui suivent la norme (un attaquant l'ignorera, bien sûr).
— La diffusion des informations NTP à tout le réseau local, pratique pour diminuer l'activité NTP (directives broadcast et broadcastclient dans le serveur NTP), ne devrait se faire que si ledit réseau est de confiance et que les informations sont authentifiées.

— Même chose pour le mode symétrique (entre pairs qui s'échangent les informations NTP, directive `peer`).

Enfin, la section 6 du RFC couvre le cas particulier des systèmes embarqués. Par exemple, les objets connectés ont une fâcheuse tendance à rester en service des années sans mise à jour. S'ils ont été configurés en usine avec une liste de serveurs NTP, et que certains de ces serveurs disparaissent ensuite, l'objet risque de ne plus pouvoir se synchroniser ou, pire, il va matraquer une machine innocente qui a récupéré l'adresse d'un serveur NTP (cf. RFC 4085). Il est donc important que les clients NTP puissent mettre à jour la liste de leurs serveurs. D'autre part, la liste doit évidemment être correcte dès le début, et ne pas inclure des serveurs NTP, même publics, sans leur autorisation. Une solution simple est de passe par le le projet « *"NTP Pool"* <http://www.pool.ntp.org/> ».

L'annexe A de notre RFC rassemble des conseils qui sont spécifiques à **une** mise en œuvre de NTP, celle de la Network Time Foundation <http://www.ntp.org/downloads.html>, le « code NTP original » (paquetage `ntp` sur Debian ou ArchLinux).

Pour obtenir une variété de sources, le démon « ntpd » fourni a la directive `pool`, qui permet de désigner un ensemble de serveurs :

```
pool 0.debian.pool.ntp.org iburst
pool 1.debian.pool.ntp.org iburst
pool 2.debian.pool.ntp.org iburst
pool 3.debian.pool.ntp.org iburst
```

NTP a la possibilité de recevoir des messages de contrôle (annexe B du RFC 1305). Cela peut être dangereux <https://www.bortzmeyer.org/ntp-reflexion.html>, et il est recommandé d'en restreindre l'accès. Avec le serveur habituel ntpd, c'est bien documenté <https://support.ntp.org/bin/view/Support/AccessRestrictions> (mais cela reste complexe et pas intuitif du tout). Voici un exemple :

```
restrict default noquery nopeer nomodify notrap
restrict ::1
restrict 2001:db8:b19:3bb0:: mask ffff:ffff:ffff:ffff:: notrust
restrict 2001:db8:2fab:e9d0:d40b:5ff:fee8:a36b nomodify
```

Dans cet exemple, la politique par défaut (première ligne) est de ne rien autoriser. Toute machine qui tenterait de parler au serveur NTP serait ignorée. Ensuite, la machine locale (`::1`, deuxième ligne) a tous les droits (aucune restriction). Entre les deux (troisième ligne), les machines du réseau local (`2001:db8:b19:3bb0::/64`) ont le droit de parler au serveur seulement si elles sont authentifiées cryptographiquement. Enfin, la machine `2001:db8:2fab:e9d0:d40b:5ff:fee8:a36b` a le droit de lire le temps chez nous mais pas de le modifier.

On avait parlé plus haut de l'importance de superviser ses services de temps. Voici une configuration avec Icinga pour faire cela :

```
apply Service "ntp-time" {
  import "generic-service"
  check_command = "ntp_time"
    assign where (host.address || host.address6) && host.vars.ntp
}
...
object Host "foobar" {
...
   vars.ntp = true
```

Avec cette configuration, la machine `foobar` sera supervisée. On peut tester depuis la ligne de commande que le "*monitoring plugin*" arrive bien à lui parler :

```
% /usr/lib/nagios/plugins/check_ntp_time -H foobar
NTP OK: Offset 7.331371307e-06 secs|offset=0.000007s;60.000000;120.000000;
```

Si la réponse avait été `CRITICAL - Socket timeout after 10 seconds`, on aurait su que le serveur refuse de nous parler.

Ah, et puisqu'on a parlé de sécurité et de protéger (un peu) NTP par la cryptographie, voici un exemple (non, ce ne sont pas mes vrais clés, je vous rassure) :

```
% cat /etc/ntp/ntp.keys
...
13 SHA1  cc5b2e7c400e778287a99b273b19dc68369922b9 # SHA1 key

% cat /etc/ntp.conf
...
keys /etc/ntp/ntp.keys
trustedkey 13
```

Avec cette configuration (le fichier `ntp.keys` peut être généré avec la commande `ntp-keygen`), le serveur NTP acceptera les messages protégés par la clé numéro 13. Sur le client, la configuration sera :

```
keys /etc/ntp/ntp.keys
server SERVERNAME key 13
```

Quelques petits trucs pour finir. Avec ntpd, comment voir l'état des pairs avec qui ont est connectés ? Ici, on a configuré l'utilisation du "*pool*" :

```
% ntpq -pn
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 0.debian.pool.n .POOL.          16 p    -   64    0    0.000    0.000   0.000
 1.debian.pool.n .POOL.          16 p    -   64    0    0.000    0.000   0.000
 2.debian.pool.n .POOL.          16 p    -   64    0    0.000    0.000   0.000
 3.debian.pool.n .POOL.          16 p    -   64    0    0.000    0.000   0.000
-162.159.200.123 10.19.11.58      3 u   25  128  377    1.381   -2.439   0.199
-164.132.45.112  43.13.124.203    3 u    8  128  377    5.507   -1.423   0.185
+212.83.145.32   193.200.43.147   2 u    4  128  377    1.047   -1.823   0.455
+5.196.160.139   145.238.203.14   2 u   12  128  377    5.000   -0.981   0.291
-163.172.61.210  145.238.203.14   2 u    8  128  377    1.037   -0.888   0.246
*82.64.45.50     .GPS.            1 u   71  128  377   11.116   -1.178   0.549
-178.249.167.0   193.190.230.65   2 u    3  128  377    6.233   -1.026   0.145
-194.57.169.1    145.238.203.14   2 u    2  128  377   10.660   -0.931   0.233
-151.80.124.104  193.204.114.232  2 u   16  128  377    4.888   -1.414   0.354
```

Autre commande utile, pour comparer l'heure locale avec les serveurs NTP :

```
%  ntpdate -q ntp.nic.fr
server 2001:67c:2218:2::4:12, stratum 2, offset 0.000520, delay 0.03194
server 2001:67c:2218:2::4:13, stratum 2, offset 0.000746, delay 0.03175
server 192.134.4.12, stratum 2, offset 0.000509, delay 0.03127
server 192.134.4.13, stratum 2, offset 0.000596, delay 0.04376
28 Oct 10:54:08 ntpdate[18996]: adjust time server 192.134.4.12 offset 0.000509 sec
```

Et avec un serveur NTP complètement différent ? Essayons avec OpenNTPD <http://www.openntpd.org/> :

```
% cat /etc/ntpd.conf

# No way to restrict per IP address :-(    Use a firewall
listen on *
servers fr.pool.ntp.org
sensor *
```

Avec cette configuration, la machine va se synchroniser au *"pool"*, cequ'on pourra vérifier avec `ntpctl` :

```
% sudo ntpctl -s all
4/4 peers valid, clock synced, stratum 4

peer
   wt tl st  next  poll          offset       delay      jitter
162.159.200.123 from pool fr.pool.ntp.org
    1 10  3   23s   30s        -2.270ms     4.795ms     0.027ms
193.52.136.2 from pool fr.pool.ntp.org
    1 10  2   32s   34s        -1.904ms    18.058ms     1.788ms
91.121.96.146 from pool fr.pool.ntp.org
 *  1 10  3    0s   31s        -1.147ms     1.872ms     0.069ms
62.210.213.21 from pool fr.pool.ntp.org
    1 10  2    1s   34s        -0.367ms     4.989ms     0.067ms
```