RFC 7384 : Security Requirements of Time Protocols in Packet Switched Networks

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 30 octobre 2014

Date de publication du RFC : Octobre 2014

https://www.bortzmeyer.org/7384.html

De plus en plus de protocoles sur l'Internet dépendent d'une horloge correcte. L'époque où les machines étaient vaguement mises à une heure approximative, de façon purement manuelle, est révolue. Aujourd'hui, il est essentiel de connaître l'heure exacte, et cela implique des dispositifs automatiques comme NTP. C'est encore plus essentiel quand il s'agit de protocoles de sécurité, comme DNSSEC (pour déterminer si une signature a expiré) ou X.509 (pour savoir si un certificat est encore valable). Mais cette utilisation du temps comme donnée dans un protocole de sécurité pose elle-même des problèmes de sécurité : si un attaquant perturbe NTP, ne risque-t-il pas d'imposer une fausse heure, et donc de subvertir des protocoles de sécurité ? D'où le groupe de travail TICTOC <https://tools.ietf.org/wg/tictoc> de l'IETF, dont voici le premier RFC : le cahier des charges des solutions de sécurité pour les protocoles de synchronisation d'horloges, comme NTP.

Deux de ces protocoles sont largement déployés aujourd'hui : un protocole IETF, NTP (RFC 5905 [1]) et PTP, alias IEEE 1588 (pas disponible en ligne, comme toutes les normes du dinosaure IEEE). NTP inclut des mécanismes de sécurité, une authentification par secret partagé (RFC 5905, notamment section 7.3) et une autre par clé publique ("*Autokey*", RFC 5906). Par contre, si PTP a un mécanisme d'authentification expérimental (annexe K de la norme), celui-ci ne semble pas avoir été formalisé complètement. Les déploiements de PTP sont donc non sécurisés.

On ne part donc pas de zéro, en tout cas pour NTP, qui a déjà une bonne partie des mécanismes demandés par ce cahier des charges. Ce RFC vise (section 1) un public composé des auteurs de logiciels de synchronisation (pour s'assurer qu'ils mettent en œuvre correctement les mécanismes de sécurité normalisés) et des auteurs de normes (qui vont devoir ajouter des mécanismes de sécurité là où ils manquent). Il peut aussi être utile aux opérationnels, lorsqu'ils révisent ou auditent la sécurité de leur

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

système de synchronisation d'horloges. À partir de questions simples, « quelles sont les menaces ? », « quels sont les coûts de la sécurité ? », « quelles sont les dépendances croisées (par exemple une authentification des serveurs de temps via un certificat dont l'usage dépend d'une horloge correcte) ? », le RFC va poser une liste d'exigences pour les solutions de sécurité.

Un petit mot sur la terminologie au passage : PTP et NTP n'ont pas le même vocabulaire mais l'analyse de notre RFC traite ensemble les deux protocoles et utilise donc un vocabulaire commun. Ainsi, les termes « maître » ("*master*") et « esclave » ("*slave*") servent pour PTP et NTP (bien qu'en NTP, on dise plutôt « serveur » et « client »). Un « grand-maître » ("*grandmaster*") est une machine qui a un accès direct à une horloge, sans passer par le protocole réseau.

Donc, commençons par l'analyse des menaces (section 3). On se base surtout sur l'article de l'auteur du RFC, T. Mizrahi, « "*Time synchronization security using IPsec and MACsec*" `<http://www.eng.tau.ac.il/~boaz/inetsem/mizrahi.pdf>` » (ISPCS 2011). Mais le RFC contient d'autres références à lire.

D'abord, une distinction entre attaquants internes et externes. Les externes sont juste connectés au même Internet que leurs victimes, ils n'ont aucun privilège particulier. Les internes, par contre, soit ont accès à une partie des clés cryptographiques utilisées, soit ont un accès au réseau sur lequel se trouvent leurs victimes et peuvent donc mener certaines attaques. Ils peuvent par exemple générer du faux trafic qui a plus de chances se sembler authentique. Se défendre contre ces Byzantins sera donc plus difficile. Si aucun mécanisme de sécurité n'est déployé, les attaquants internes et externes ont les mêmes possibilités.

Autre distinction importante entre les attaquants, ceux capables d'être Homme du Milieu et les autres. Un homme du milieu est placé de telle façon qu'il peut intercepter les paquets et les modifier. Les autres peuvent injecter des paquets (y compris en rejouant des paquets qu'ils ont lu) mais pas modifier le trafic existant.

Maintenant, quelles sont les attaques possibles ? La section 3.2 en fournit une longue liste. Rappelez-vous que ce sont des attaques théoriques : beaucoup sont déjà empêchées par les techniques de sécurité existant dans NTP (si elles sont activées...) Un homme du milieu peut modifier les paquets, en changeant les informations sur le temps, donnant ainsi de fausses informations. Tout va sembler OK mais les ordinateurs vont s'éloigner de l'heure réelle. Un attaquant actif peut aussi fabriquer, en partant de zéro, un faux paquet, contenant ces fausses informations. Il peut se faire passer pour le maître, trompant ainsi les esclaves, ou pour un esclave, donnant des informations au maître qui vont lui faire envoyer une réponse erronée au vrai esclave. Là aussi, tout semblera marcher, les horloges seront synchronisées, mais l'information sera fausse. Si l'attaquant a du mal à modifier des paquets ou à les générer (par exemple en raison d'une mesure de sécurité), il peut aussi tenter de rejouer des paquets du passé qu'il aura copiés. Cela permet potentiellement de faire reculer les horloges des victimes.

Un autre moyen de se faire accepter comme le maître, au lieu d'usurper l'identité du vrai maître, est de tricher dans le processus de choix du maître. Par exemple, en PTP, le maître est choisi par un algorithme nommé BMCA ("*Best Master Clock Algorithm*") et le tricheur peut donc arriver à se faire désigner comme maître (rappelez-vous que PTP n'a guère de sécurité).

Un attaquant homme du milieu peut aussi jeter les paquets (faisant croire qu'un maître n'est pas joignable) ou les retarder délibérement : ce retard peut permettre de modifier l'horloge, puisque les participants au protocole NTP utilisent le temps d'aller-retour pour calculer l'heure chez le pair. Certaines protections anti-rejeu (comme le fait de vérifier qu'on ne reçoit pas de copie d'un paquet déjà reçu) ne marchent pas contre cette attaque.

Certaines de ces attaques peuvent se faire par des moyens très divers, situés dans la couche 2 ou dans la couche 3 : "*ARP spoofing*", par exemple.

Une solution évidente à bien des problèmes de sécurité est la cryptographie. Mais attention, elle ouvre la voie à de nouvelles attaques comme l'envoi de paquets mal formés qui mènent à des calculs cryptographiques très longs, réalisant ainsi une attaque par déni de service. Une autre attaque par déni de service moins sophistiquée serait de noyer la machine sous un volume énorme de paquets NTP ou PTP.

L'attaquant peut aussi viser le grand-maître : si ce dernier utilise le GPS, un attaquant qui peut envoyer des faux signaux GPS peut tromper le grand-maître, trompant en cascade tous les autres participants. Aucune défense située dans le protocole de synchronisation ne sera alors effective, puisque l'attaque a lieu en dehors de ce protocole. Elle échappe donc au groupe de travail TICTOC.

Bien sûr, il y a aussi des menaces exploitant une faille des programmes qui mettent en œuvre les protocoles, plutôt que des protocoles eux-mêmes, par exemple les attaques par réflexion `<https://www.bortzmeyer.org/ntp-reflexion.html>` utilisant NTP, lorsque le serveur accepte trop facilement des requêtes normalement utiles uniquement pour le déboguage.

Plus sophistiquées (peut-être trop pour être utilisées en vrai), les reconnaissances d'un réseau conduites grâce aux protocoles de synchronisation d'horloge. Un attaquant, pour préparer de futures attaques, commence par reconnaître sa cible, chercher les machines, etc. Avec la synchronisation d'horloges, il peut, passivement, collecter des informations qui, par leur fréquence et leur contenu, permettent, entre autres, de "*fingerprinter*" des machines spécifiques.

Les attaques possibles sont résumées en section 3.3, avec un tableau qui indique pour chaque attaque l'**impact** (horloges complètement faussées, précision diminuée - mais l'horloge reste proche du temps réel, ou déni de service) et le **type d'attaquant** qui peut les effectuer (interne ou externe, devant être homme du milieu ou pas).

De cette liste d'attaques et de leurs caractéristiques, notre RFC va tirer un certain nombre d'exigences auxquelles devront se conformer les solutions de sécurité. Chacune de ces exigences aura un niveau d'impérativité (les "*MUST*" et "*SHOULD*" du RFC 2119) selon l'impact et selon le type d'attaquant. Par exemple, les attaques qui peuvent être faites par un attaquant externe sont plus graves (le nombre d'attaquants potentiels est plus élevé) et ont donc plus souvent un "*MUST*" (section 4 pour la discussion sur les niveaux). Notez que la solution n'est pas forcément interne au protocole de synchronisation d'horloges : utiliser IPsec ou IEEE 802.1AE est souvent une bonne solution.

La section 5 donne la liste des exigences elle-mêmes. La première est l'authentification : il FAUT un mécanisme d'authentification (vérifier que la machine est bien ce qu'elle prétend être) et d'autorisation (vérifier que la machine qui annonce être un maître en a le droit). Notamment, les esclaves doivent pouvoir vérifier les maîtres. Ces mécanismes doivent être récursifs, car les protocoles de synchronisation d'horloges passent souvent par des relais intermédiaires (en NTP, une machine de strate 3 ne reçoit pas directement l'horloge, mais le fait via la machine de strate 2 - le RFC 5906 avait inventé le terme de "*proventication*", contraction de "*provenance*" et "*authentication*" pour décrire cette question). Cette authentification permet de répondre aux attaques par usurpation d'identité. Sans cela, des attaques triviales sont possibles.

Par contre, l'authentification des esclaves par les maîtres est facultative ("*MAY*") car un esclave malveillant ne peut pas tellement faire de mal à un maître (à part peut-être le déni de service par l'afflux

d'esclaves non autorisés) et qu'un esclave a peu de maîtres alors qu'un maître a beaucoup d'esclaves, dont l'authentification pourrait nécessiter trop de travail.

Le RFC impose aussi l'existence d'un mode « fermé » où seuls des pairs authentifiés peuvent participer (il existe sur l'Internet beaucoup de serveurs NTP publics, acceptant des connexions de la part d'inconnus).

Le RFC exige un mécanisme de protection contre les attaques d'un homme du milieu essayant de diminuer la qualité des données, par exemple en retardant délibérement des paquets. En NTP, c'est par exemple le fait d'avoir plusieurs maîtres.

Après l'authentification (et l'autorisation), l'intégrité des paquets : notre RFC exige un mécanisme d'intégrité, permettant de s'assurer qu'un méchant n'a pas modifié les paquets en cours de route. Cela se fait typiquement par un ICV ("*Integrity Check Value*"). La vérification de l'intégrité peut se faire à chaque relais (ce qui est plus simple), ou bien de bout en bout (ce qui protège si un des relais est un vilain Byzantin).

L'usurpation d'identité est ensuite couverte : le RFC exige ("*MUST*") un mécanisme permettant d'empêcher un attaquant de se faire passer pour le maître. Comme son nom l'indique, le maître peut modifier les horloges des autres machines à volonté et empêcher une usurpation est donc critique. Le RFC impose en outre un mécanisme contre le rejeu (car ce sont des attaques triviales à monter) et un moyen de changer les clés cryptographiques utilisées (cela a l'air évident mais certains protocoles - comme OSPF, cf. RFC 6039 - sont très pénibles pour cela, et en pratique les clés ne sont pas changées).

Les demandes suivantes sont moins fortes, car concernant des attaques moins graves ou plus difficiles à faire. Par exemple, il faudrait, si possible ("*SHOULD*"), fournir des mécanismes contre les attaques par déni de service.

Enfin viennent les simples suggestions, comme de prévoir un mécanisme assurant la confidentialité des paquets. Comme aucune information vraiment secrète ne circule dans les messages de synchronisation d'horloges, cette suggestion est un simple "*MAY*".

Ces exigences sont résumées dans la section 6, sous la forme d'un tableau synthétique.

La section 7 rassemble un ensemble de questions diverses sur la sécurité des protocoles de synchronisation d'horloge. Par exemple, la section 7.4 se penche sur les interactions entre cette synchronisation et des protocoles extérieurs de sécurité comme IPsec. La 7.5 expose un cas plus embêtant, le problème d'œuf et de poule entre la synchronisation d'horloges et des protocoles qui utilisent l'heure dans leurs mécanismes de sécurité. Par exemple, si on utilise TLS avec authentification X.509 à un moment quelconque pour se connecter, une horloge correcte est nécessaire (vérification de la non-expiration du certificat), alors que NTP n'est pas forcément déjà disponible. Il faut donc une horloge stable, pour ne pas trop dériver en attendant que NTP prenne le relais.

Enfin, la section 8 rappelle que le problème difficile de la distribution des clés n'est pas encore traité, que ce soit dans ce RFC ou dans les protocoles.

Voici un court exemple de sécurisation avec NTP. Je n'ai pas utilisé la technique "*Autokey*" du RFC 5906 mais la plus ancienne (voire archaïque), qui utilise des clés privées. L'un des avantages est la portabilité, on sait que cela va marcher sur tous les clients et serveurs NTP. Sur le serveur, je crée les clés :

```
% sudo ntp-keygen  -M
Built against OpenSSL OpenSSL 1.0.1i 6 Aug 2014, using version OpenSSL 1.0.1j 15 Oct 2014
Generating new md5 file and link
ntpkey_md5_alarmpi->ntpkey_MD5key_alarmpi.3623647899

% sudo ln -s ntpkey_md5_alarmpi ntp.keys

% cat ntp.keys
...
11 SHA1 2db9278081f6b410cfb826317b87cf95d5b90689  # SHA1 key
12 SHA1 cc52f97bc43b6c43a4cbe446813f0c6c3bd54f7c  # SHA1 key
```

Et j'indique dans la configuration que je n'autorise (`notrust`) que les clients du réseau local, et s'ils s'authentifient avec la clé n° 11 :

```
keys /etc/ntp/ntp.keys
trustedkey 11
...
restrict default noquery nopeer nomodify notrap
...
restrict 2a01:e35:8bd9:8bb0:: mask ffff:ffff:ffff:ffff:: notrust
```

Avec une telle restriction, les clients qui ne s'authentifient pas ne reçoivent pas de réponse (même pas une réponse négative), ce qui rend les problèmes d'authentification difficiles à déboguer, d'autant plus que Wireshark n'indique apparemment nulle part si les paquets sont authentifiés ou pas. Côté client, il faut donc montrer patte blanche, d'abord copier les clés puis :

```
keys /etc/ntp/ntp.keys
trustedkey 11

server my-favorite-server key 11
```

Et, cette fois, le serveur répond et tout se passe bien.

Si vous voulez faire de l'"*Autokey*" (cryptographie à clé publique), vous pouvez consulter la documentation officielle <http://support.ntp.org/bin/view/Support/ConfiguringAutokey> ou bien, en français, l'article de Laurent Archambault <http://archil.fr/Autokey-IFF.xhtml> (ou, en général, son cours NTP <http://archi.laurent.perso.neuf.fr/Doc_reseauNTP.html>). Si, au lieu d'être sur Unix avec la mise en œuvre de référence ntpd, vous gérez un routeur Cisco, voyez en français cet article <http://www.lolokai.com/blog/2011/11/17/cisco-et-le-ntp-synchronisez-les-hor>. Si vous utilisez le programme des gens d'OpenBSD, OpenNTPD <http://www.openntpd.org>, tant pis pour vous, il n'a aucun mécanisme de sécurité. Je n'ai pas trouvé de documents ANSSI sur la sécurisation des protocoles de synchronisation d'horloge. Le NIST a un service NTP authentifié <http://www.nist.gov/pml/div688/grp40/auth-ntp.cfm> (notez la complexité des procédures d'inscription !). Pendant longtemps, la référence en français était l'excellent article d'Alain Thivillon <http://www.hsc.fr/ressources/breves/ntp-auth.html.fr> mais il est trop ancien aujourd'hui.