# RFC 8315 : Cancel-Locks in Netnews articles

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 14 février 2018

Date de publication du RFC : Février 2018

http://www.bortzmeyer.org/8315.html

---

Cela peut sembler étonnant, mais le service des "*News*" fonctionne toujours. Et il est régulièrement perfectionné. Ce nouveau RFC normalise une extension au format des articles qui permettra de sécuriser un petit peu l'opération d'annulation d'articles.

Une fois qu'un article est lancé sur un réseau social décentralisé, comme Usenet (RFC 5537 [1]), que faire si on le regrette ? Dans un système centralisé comme Twitter, c'est simple, on s'authentifie, on le supprime et plus personne ne le voit. Mais dans un réseau décentralisé, il faut encore propager la demande de suppression (d'annulation, sur les "*News*"). Et cela pose évidemment des questions de sécurité : il ne faut pas permettre aux méchants de fabriquer trop facilement une demande d'annulation. Notre RFC propose donc une mesure de sécurité, l'en-tête `Cancel-Lock:`.

Cette mesure de sécurité est simple, et ne fournit certainement pas une sécurité de niveau militaire. Pour la comprendre, il faut revenir au mécanisme d'annulation d'un article d'Usenet. Un article de contrôle est un article comme les autres, envoyé sur Usenet, mais il ne vise pas à être lu par les humains, mais à être interprété par le logiciel. Un exemple d'article de contrôle est l'article de contrôle d'annulation, défini dans le RFC 5337, section 5.3. Comme son nom l'indique, il demande la suppression d'un article, identifié par son "*Message ID*". Au début d'Usenet, ces messages de contrôle n'avaient aucune forme d'authentification. On a donc vu apparaitre des faux messages de contrôle, par exemple à des fins de censure (supprimer un article qu'on n'aimait pas). Notre nouveau RFC propose donc qu'un logiciel proche de la source du message mette un en-tête `Cancel-Lock:` qui indique la clé qui aura le droit d'annuler le message.

Évidemment, ce `Cancel-Lock:` n'apporte pas beaucoup de sécurité, entre autre parce qu'un serveur peut toujours le retirer et mettre le sien, avant de redistribuer (c'est évidemment explicitement interdit par le RFC mais il y a des méchants). Mais cela ne change de toute façon pas grand'chose à la situation

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc5537.txt`

actuelle, un serveur peut toujours jeter un article, de toute façon. Si on veut quand même une solution de sécurité « sérieuse », il faut utiliser PGP, comme mentionné en passant par le RFC 5537 (mais jamais normalisé <https://datatracker.ietf.org/doc/draft-lindsey-usefor-signed/> dans un RFC).

La section 2 du RFC décrit en détail le mécanisme de sécurité. La valeur de l'en-tête `Cancel-Lock:` est l'encodage en base64 d'une condensation d'une clé secrète (en fait, on devrait plutôt l'appeler mot de passe). Pour authentifier une annulation, le message de contrôle comportera un autre en-tête, `Cancel-Key:`, qui révélera la clé (qui ne devra donc être utilisée qu'une fois).

Voici un exemple. On indique explicitement l'algorithme de condensation (ici, SHA-256, la liste est dans un registre IANA <https://www.iana.org/assignments/netnews-parameters/netnews-parameters.xml#cancel-lock-hash-algorithms>). D'abord, le message original aura :

```
Cancel-Lock: sha256:s/pmK/3grrz++29ce2/mQydzJuc7iqHn1nqcJiQTPMc=
```

Et voici le message de contrôle, authentifié :

```
Cancel-Key: sha256:qv1VXHYiCGjkX/N1nhfYKcAeUn8bCVhrWhoKuBSnpMA=
```

La section 3 du RFC détaille comment on utilise ces en-têtes. Le `Cancel-Lock:` peut être mis par l'auteur originel de l'article, ou bien par un intermédiaire (par exemple le modérateur qui a approuvé l'article). Plusieurs `Cancel-Lock:` peuvent donc être présents. Notez qu'il n'y a aucun moyen de savoir si le `Cancel-Lock:` est « authentique ». Ce mécanisme est une solution de sécurité faible.

Pour annuler un message, on envoie un message de contrôle avec un `Cancel-Key:` correspondant à un des `Cancel-Lock:`. Les serveurs recevant ce message de contrôle condenseront la clé (le mot de passe) et vérifieront s'ils retombent bien sur le condensat contenu dans un des `Cancel-Lock:`.

La section 4 donne les détails sur le choix de la clé (du mot de passe). Évidemment, elle doit être difficile à deviner, donc plutôt choisie par un algorithme pseudo-aléatoire (et pas "azerty123"). Et elle doit être à usage unique puisque, une fois révélée par un `Cancel-Key:`, elle n'est plus secrète. L'algorithme recommandé par le RFC est d'utiliser un HMAC (RFC 2104) d'un secret et de la concaténation du "*Message ID*" du message avec le nom de l'utilisateur. Comme cela, générer un `Cancel-Key:` pour un message donné peut se faire avec juste le message, sans avoir besoin de mémoriser les clés. Voici un exemple, tiré de la section 5, et utilisant OpenSSL. Le secret est `frobnicateZ32`. Le message est le `<12345@example.net>` et l'utilisateur est `stephane` :

```
% printf "%s" "<12345@example.net>stephane" \
        | openssl dgst -sha256 -hmac "frobnicateZ32" -binary \
        | openssl enc -base64
f0rHwfZXp5iKFjTbX/I5bQXh9Dta33nWBzLi8f9oaoM=
```

Voilà, nous avons notre clé, `f0rHwfZXp5iKFjTbX/I5bQXh9Dta33nWBzLi8f9oaoM=`. Pour le condensat, nous nous servirons de SHA-256 :

```
% printf "%s" "f0rHwfZXp5iKFjTbX/I5bQXh9Dta33nWBzLi8f9oaoM=" \
        | openssl dgst -sha256 -binary \
        | openssl enc -base64
RBJ8ZsgqBnW/tYT/qu1JcXK8SA2O9g+qJLDzRY5h1cg=
```

Nous pouvons alors former le `Cancel-Lock:` :

```
Cancel-Lock: sha256:RBJ8ZsgqBnW/tYT/qu1JcXK8SA2O9g+qJLDzRY5h1cg=
```

Et, si nous voulons annuler ce message, le `Cancel-Key:` dans le message de contrôle d'annulation aura l'air :

```
Control: cancel <12345@example.net>
Cancel-Key: sha256:f0rHwfZXp5iKFjTbX/I5bQXh9Dta33nWBzLi8f9oaoM=
```

Pour vérifier ce message de contrôle, le logiciel calculera le condensat de la clé et vérifiera s'il retombe bien sur `RBJ8ZsgqBnW/tYT/qu1JcXK8SA2O9g+qJLDzRY5h1cg=`.

Enfin, la section 7 du RFC détaille la sécurité de ce mécanisme. Cette sécurité est plutôt faible :

— Aucune protection de l'intégrité. Un intermédiaire a pu modifier, ajouter ou supprimer le `Control-Lock:`. Si cela vous défrise, vous devez utiliser PGP.
— Lors d'une annulation, la clé est visible par tous, donc elle peut être copiée et utilisée dans un message de contrôle de remplacement (au lieu de l'annulation). Mais, de toute façon, un attaquant peut aussi bien faire un nouveau message faux (Usenet ne vérifie pas grand'chose).
— Avant ce RFC, ce mécanisme était déjà largement utilisé, depuis longtemps, et souvent en utilisant SHA-1 comme fonction de condensation. Contrairement à ce qu'on lit parfois, SHA-1 n'est pas complètement cassé : il n'y a pas encore eu de publication d'une attaque pré-image pour SHA-1 (seulement des collisions). Néanmoins, SHA-1 ne peut plus être considéré comme sûr, d'autant plus que Usenet évolue très lentement : un logiciel fait aujourd'hui va rester en production longtemps et, avec le temps, d'autres attaques contre SHA-1 apparaitront. D'où la recommandation du RFC de n'utiliser que SHA-2.

Les deux nouveaux en-têtes ont été ajoutés au registre des en-têtes <https://www.iana.org/assignments/message-headers/message-headers.xml#perm-headers>.

À noter que, comme il n'y a pas de groupe de travail IETF sur les "*News*", ce RFC a été surtout discuté…sur les "*News*", en l'occurrence les groupes `news.software.nntp` et `de.comm.software.newsserver`. Comme ce RFC décrit une technique ancienne, il y a déjà de nombreuses mises en œuvre comme la bibliothèque canlock <http://albasani.net/wiki/Cancel-Lock> (paquetage Debian libcanlock2), dans le serveur INN <http://svn.schnuerpel.eu/viewvc.cgi/trunk/lib/Schnuerpel/INN/?root=schnuerpel>, ou les clients "*News*" Gnus (regardez cet article sur son usage <https://unix.stackexchange.com/questions/59635/canlock-password-hashed-password-mysteriously-in-e
59990>), flnews <http://micha.freeshell.org/flnews/>, slrn ou tin. Vous pouvez aussi lire l'article de Brad Templeton <http://www.templetons.com/usenet-format/howcancel.html> comparant `Cancel-Lock:` aux signatures.

Merci à Julien Élie pour sa relecture.