RFC 8967 : Message Authentication Code (MAC) Authentication for the Babel Routing Protocol

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 12 janvier 2021

Date de publication du RFC : Janvier 2021

https://www.bortzmeyer.org/8967.html

Le protocole de routage Babel, normalisé dans le RFC 8966 [1], avait à l'origine zéro sécurité. N'importe quelle machine pouvait se dire routeur et annoncer ce qu'elle voulait. Le RFC 7298 avait introduit un mécanisme d'authentification. Ce nouveau RFC le remplace avec un nouveau mécanisme. Chaque paquet est authentifié par HMAC, et les routeurs doivent donc partager une clé secrète.

Comme, par défaut, Babel croit aveuglément ce que ses voisins lui racontent, un méchant peut facilement, par exemple, détourner du trafic (en annonçant une route de plus forte préférence). Le problème est bien connu mais n'a pas de solution simple. Par exemple, Babel permet à la fois de l'"*unicast*" et du "*multicast*", le second étant plus difficile à protéger. Une solution de sécurité, DTLS, spécifiée pour Babel dans le RFC 8968, résout le problème en ne faisant que de l'"*unicast*". Notre RFC choisit une autre solution, qui marche pour l'"*unicast*" et le "*multicast*". Chaque paquet est authentifié par un MAC attaché au paquet, calculé entre autres à partir d'une clé partagée.

La solution décrite dans ce RFC implique que **tous** les routeurs connectés au réseau partagent la clé secrète, et que tous ces routeurs soient eux-mêmes de confiance (l'authentification n'implique pas l'honnêteté du routeur authentifié, point très souvent oublié quand on parle d'authentification…) En pratique, c'est un objectif difficile à atteindre, il nécessite un réseau relativement bien géré. Pour un rassemblement temporaire où tout le monde partage sa connectivité, faire circuler ce mot de passe partagé sera difficile.

Ce mécanisme a par contre l'avantage de ne pas nécessiter que les horloges soient correctes ou synchronisées, et ne nécessite pas de stockage de données permanent (ce qui serait contraignant pour, par exemple, certains objets connectés).

Pour que tout marche bien et qu'on soit heureux et en sécurité, ce mécanisme d'authentification compte sur deux pré-requis :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc8966.txt

— Une fonction MAC sécurisée (évidemment), c'est-à-dire qu'on ne puisse pas fabriquer un MAC correct sans connaitre la clé secrète,
— Que les nœuds Babel arrivent à ne pas répéter certaines valeurs qu'ils transmettent. Cela peut se faire avec un générateur aléatoire correct (RFC 4086) mais il existe aussi d'autres solutions pour générer des valeurs uniques, non réutilisables.

En échange de ces garanties, le mécanisme de ce RFC garantit l'intégrité des paquets et le rejet des rejeux (dans certaines conditions, voir le RFC).

La section 2 du RFC résume très bien le protocole : quand un routeur Babel envoie un paquet sur une des interfaces où la protection MAC a été activée, il calcule le MAC et l'ajoute à la fin du paquet. Quand il reçoit un paquet sur une des interfaces où la protection MAC a été activée, il calcule le MAC et, s'il ne correspond pas à ce qu'il trouve à la fin du paquet, le paquet est jeté. Simple, non ? Mais c'est en fait un peu plus compliqué. Pour protéger contre les attaques par rejeu, la machine qui émet doit maintenir un compteur des paquets envoyés, le PC ("*Packet Counter*"). Il est inclus dans les paquets envoyés et protégé par le MAC.

Ce PC ne protège pas dans tous les cas. Par exemple, si un routeur Babel vient de démarrer, et n'a pas de stockage permanent, il ne connait pas les PC de ses voisins et ne sait donc pas à quoi s'attendre. Dans ce cas, il doit ignorer le paquet et mettre l'émetteur au défi de répondre à un numnique <https://www.bortzmeyer.org/nonce.html> qu'il envoie. Le voisin répond en incluant le numnique et son nouveau PC, prouvant ainsi qu'il ne s'agit pas d'un rejeu.

Petite difficulté, en l'absence de stockage permanent, le PC peut revenir en arrière et un PC être réutilisé. Outre le PC, il faut donc un autre nombre, l'index. Celui-ci n'est, comme le numnique utilisé dans les défis, jamais réutilisé. En pratique, un générateur aléatoire est une solution raisonnable pour fabriquer numniques et index.

La section 3 du RFC décrit les structures de données qu'il faut utiliser pour mettre en œuvre ce protocole. La table des interfaces (RFC 8966, section 3.2.3), doit être enrichie avec une indication de l'activation de la protection MAC sur l'interface, et la liste des clés à utiliser (Babel permet d'avoir plusieurs clés, notamment pour permettre le remplacement d'une clé, et le récepteur doit donc les tester toutes). Il faut également ajouter à la table des interfaces l'index et le PC décrits plus haut.

Quant à la table des (routeurs) voisins (RFC 8966, section 3.2.4), il faut y ajouter l'index et le PC de chaque voisin, et le numnique.

Enfin la section 4 détaille le fonctionnement. Comment calculer le MAC (avec un pseudo-en-tête), où mettre le TLV qui indique le PC, et celui qui contient la ou les MAC (dans la remorque, c'est-à-dire la partie du paquet après la longueur explicite), etc. La section 6 fournit quant à elle le format exact des TLV utilisés : le MAC TLV qui stocke le MAC, le PC TLV qui indique le compteur, et les deux TLV qui permettent de lancer un défi et d'obtenir une réponse, pour se resynchroniser en cas d'oubli du compteur. Ils ont été ajoutés au registre IANA <https://www.iana.org/assignments/babel/babel.xml#tlv-types>.

Les gens de l'opérationnel aimeront la section 5, qui décrit comment déployer initialement cette option, et comment effectuer un changement de clé. Pour le déploiement initial, il faut configurer les machines dans un mode où elles signent mais ne vérifient pas les paquets entrants (il faut donc que les implémentations aient un tel mode). Cela permet de déployer progressivement. Une fois tous les routeurs ainsi configurés, on peut activer le mode normal, avec signature **et** vérification. Pour le remplacement de clés, on ajoute d'abord la nouvelle clé aux routeurs, qui signent donc avec les deux clés, l'ancienne et la nouvelle, puis, une fois que tous les routeurs ont la nouvelle clé, on retire l'ancienne.

Un petit bilan de la sécurité de ce mécanisme, en section 7 : d'abord un rappel qu'il est simple et qu'il ne fournit que le minimum, l'authentification et l'intégrité des paquets. Si on veut d'avantage, il faut passer à DTLS, avec le RFC 8968. Ensuite, certaines valeurs utilisées doivent être générées sans que l'attaquant puisse les deviner, ce qui nécessite, par exemple, un générateur de nombres aléatoires sérieux. D'autant plus que la taille limitée de certaines valeurs peut permettre des attaques à la force brute. Si, par exemple, les clés dérivent d'une phrase de passe, il faut une bonne phrase de passe, plus une fonction de dérivation qui gêne ces attaques, comme PBKDF2 (RFC 2898), bcrypt ou scrypt (RFC 7914).

Et, comme toujours, il y a la lancinante menace des attaques par déni de service, par exemple en envoyant des paquets délibérement conçus pour faire faire des calculs cryptographiques inutiles aux victimes. C'est pour limiter leurs conséquences que, par exemple, Babel impose de limiter le rythme d'envoi des défis, pour éviter de s'épuiser à défier un attaquant.

Je n'ai pas vu quelles mises en œuvre de Babel avaient ce mécanisme. il ne semble pas dans la version officielle de babeld <https://www.irif.fr/~jch/software/babel/#download>, en tout cas, même si il y a du code dans une branche séparée <https://github.com/jech/babeld/tree/hmac>. Et ce n'est pas encore dans une version officielle de BIRD <http://bird.network.cz/> mais le code est déjà écrit.

Le RFC ne décrit pas les changements depuis le RFC 7298 car il s'agit en fait d'un protocole différent, et incompatible (ce ne sont pas les mêmes TLV, par exemple).

Merci à Juliusz Chroboczek pour sa relecture.