RFC 8547 : TCP-ENO: Encryption Negotiation Option

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 23 mai 2019

Date de publication du RFC : Mai 2019

https://www.bortzmeyer.org/8547.html

Ce RFC, tout juste sorti des presses, décrit une extension de TCP nommée ENO, pour "*Encryption Negotiation Option*". Elle permet d'indiquer qu'on souhaite chiffrer la communication avec son partenaire TCP, et de négocier les options. Elle sert au protocole tcpcrypt, décrit, lui, dans le RFC 8548 [1].

Malgré le caractère massif de la surveillance exercée sur les communications Internet, il y a encore des connexions TCP dont le contenu n'est pas chiffré. Cela peut être parce que le protocole applicatif ne fournit pas de moyen (genre une commande `STARTTLS`) pour indiquer le passage en mode chiffré, ou simplement parce que les applications ne sont plus guère maintenues et que personne n'a envie de faire le travail pour, par exemple, utiliser TLS. Pensons à whois (RFC 3912), par exemple. La nouvelle option ENO va permettre de chiffrer ces protocoles et ces applications, en agissant uniquement dans la couche transport, au niveau TCP.

Le but de cette nouvelle option TCP est de permettre aux deux pairs TCP de se mettre d'accord sur le fait d'utiliser le chiffrement, et quel chiffrement. Ensuite, tcpcrypt (RFC 8548) ou un autre protocole utilisera cet accord pour chiffrer la communication. En cas de désaccord, on se rabattra sur du TCP « normal », en clair.

Le gros de la spécification est dans la section 4 du RFC. ENO est une option TCP (cf. RFC 793, section 3.1). Elle porte le numéro 69 (qui avait déjà été utilisé par des protocoles analogues mais qui étaient restés encore plus expérimentaux) et figure dans le registre des options TCP <https://www.iana.org/assignments/tcp-parameters/tcp-parameters.xml#tcp-parameters-1>. (0x454E a été gardé pour des expériences, cf. RFC 6994.) Le fait d'envoyer cette option indique qu'on veut du chiffrement. Chaque possibilité de chiffrement, les TEP ("*TCP Encryption Protocol*") est dans une sous-option de l'option ENO (section 4.1 pour les détails de format). Si la négociation a été un succès, un TEP est choisi.

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc8548.txt`

Les TEP sont décrits dans d'autres RFC (par exemple, le RFC 8548, sur tcpcrypt, en décrit quatre). Les TEP sont enregistrés à l'IANA <https://www.iana.org/assignments/tcp-parameters/tcp-parameters.xml#tcp-encryption>.

À noter que TCP est symétrique : il n'y a pas de « client » ou de « serveur », les deux pairs peuvent entamer la connexion simultanément (BGP, par exemple, utilise beaucoup cette possibilité). ENO, par contre, voit une asymétrie : les deux machines qui communiquent sont nommées A et B et ont des rôles différents.

A priori, c'est A qui enverra un SYN (message de demande d'établissement de connexion). Ce SYN inclura l'option ENO, et ce sera de même pour les trois messages de la triple poignée de mains TCP. La section 6 du RFC donne quelques exemples. Ainsi :

— A ¿ B : SYN avec ENO (X, Y) - TEP X et Y,
— A ¡ B : SYN+ACK avec ENO (Y)
— A ¿ B : ACK avec ENO vide

Cet échange mènera à un chiffrement fait avec le TEP Y, le seul que A et B avaient en commun. Par contre, si B est un vieux TCP qui ne connait pas ENO :

— A ¿ B : SYN avec ENO (X, Y) - TEP X et Y,
— A ¡ B : SYN+ACK sans ENO
— A ¿ B : ACK sans ENO

Ne voyant pas de ENO dans le SYN+ACK, A renonce au chiffrement. La connexion TCP ne sera pas protégée.

Et les TEP ("*TCP Encryption Protocol*"), qu'est-ce qu'ils doivent définir ? La section 5 détaille les exigences pour ces protocols. Notamment :

— Ils doivent chiffrer (évidemment) avec un algorithme de chiffrement intègre (cf. RFC 5116),
— définir un "*session ID*", un identificateur de session unique et imprévisible (pour les applications qui souhaiteraient faire leur authentification et la lier à une session particulière),
— ne pas accepter d'algorithmes de chiffrement trop faibles, ou, bien sûr, nuls (cela parait drôle mais certaines protocoles autorisaient explicitement un chiffrement sans effet),
— fournir de la confidentialité persistante.

Si, à ce stade, vous vous posez des questions sur les choix faits par les concepteurs d'ENO, et que vous vous demandez pourquoi diable ont-ils décidé ceci ou cela, il est temps de lire la section 8, qui explique certains choix de conception. D'abord, une décision importante était qu'en cas de problème lors de la négociation, la connexion devait se replier sur du TCP classique, non chiffré, et surtout ne pas échouer. En effet, si un problème de négociation empêchait la connexion de s'établir, personne n'essayerait d'utiliser ENO. Donc, si on n'a pas d'option ENO dans les paquets qu'on reçoit, on n'insiste pas, on repasse en TCP classique. Et ceci, que les options n'aient jamais été mises, ou bien qu'elles aient été retirées par un intermédiaire trop zélé. On trouve de tout dans ces machines intermédiaires, y compris les comportements les plus délirants. Le RFC note ainsi que certains répartiteurs de charge renvoient à l'expéditeur les options TCP inconnues. L'émetteur pourrait alors croire à tort que son correspondant accepte ENO même quand ce n'est pas vrai. Un bit nommé b, mis à 0 par la machine A et à 1 par la machine B, permet de détecter ce problème, et de ne pas tenter de chiffrer avec un correspondant qui ne sait pas faire.

Cette asymétrie (une machine met le bit b à 1 mais pas l'autre) est un peu ennuyeuse, car TCP est normalement symétrique (deux machines peuvent participer à une ouverture simultanée de connexion, cf. RFC 793, section 3.4). Mais aucune meilleure solution n'a été trouvée, d'autant plus qu'une machine ne sait qu'il y a eu ouverture simultanée qu'après avoir envoyé son SYN (et si le message SYN de l'autre machine est perdu, elle ne saura jamais qu'une ouverture simultanée a été tentée).

Les protocoles utilisant ENO, comme tcpcrypt, sont conçus pour fonctionner sans la participation de l'application. Mais si celle-ci le souhaite, elle peut s'informer de l'état de sécurisation de la connexion

TCP, par exemple pour débrayer un chiffrement au niveau applicatif, qui n'est plus nécessaire. Le bit a dans l'option ENO sert à cela. Mis à 1 par une application, il sert à informer l'application en face qu'on peut tenir compte du chiffrement, par exemple pour activer des services qui ont besoin d'une connexion sécurisée. (Notez qu'il n'existe pas d'API standard pour lire et modifier le bit a, ce qui limite les possibilités.)

La section 7 de notre RFC explique quelques développements futurs qui pourraient avoir lieu si des améliorations futures à TCP se répandent. Ainsi, si de nouvelles API, plus perfectionnées que celles du RFC 3493, permettent à TCP de connaitre non seulement l'adresse IP de la machine où on veut se connecter mais également son nom, on pourrait imaginer une authentification fondée sur le nom, par exemple avec DANE (RFC 6394). On pourrait aussi imaginer qu'ENO permette de sélectionner et de démarrer TLS même sans que l'application soit au courant.

Dans l'Internet très ossifié d'aujourd'hui, il est difficile de déployer quelque chose de nouveau, comme l'option ENO (d'où le statut expérimental de ce RFC.) On risque toujours de tomber sur un intermédiaire qui se croit autorisé à modifier ou jeter des paquets dont la tête ne lui revient pas. La section 9 du RFC analyse deux risques :

— Le rique de repasser en TCP classique, non chiffré, par exemple si un intermédiaire supprime l'option ENO,
— le risque de ne pas pouvoir se connecter du tout, par exemple si un intermédiaire jette les paquets contenant l'option ENO.

Le premier risque n'est pas trop sérieux, ENO était prévu pour du déploiement incrémental, de toute façon (on ne peut pas espérer que toutes les machines adoptent ENO en même temps.) Le deuxième est plus grave et, s'il s'avère trop commun, il faudra des heuristiques du genre « si pas de réponse en N millisecondes, réessayer sans ENO ».

Outre ces risques, il est toujours possible, lorsqu'on touche à un protocole aussi crucial que TCP, que d'autres choses aillent mal, et il est donc nécessaire d'expérimenter. Il y a aussi des inconnues du genre « les applications vont-elles tirer profit d'ENO ? » (ce qui n'est pas nécessaire mais pourrait être utile).

La section 10 du RFC étudie les questions de sécurité soulevées par ENO. Comme ENO vise à permettre, entre autres, le chiffrement opportuniste (on chiffre si on peut, sinon on passe en clair, et on n'impose pas d'authentification, cf. RFC 7435), il faut être bien conscient des limites de ce modèle. Le chiffrement opportuniste protège bien contre un surveillant purement passif, mais pas contre un attaquant actif qui pourrait, par exemple, supprimer toutes les options ENO des paquets TCP, ou bien se mettre en position de terminaison TCP, avant de relayer vers le vrai destinataire, agissant ainsi en homme du milieu. Il ne faudrait donc pas prétendre à l'utilisateur que sa connexion est sûre.

Une solution est l'authentification, et c'est bien à cela que sert le "*session ID*". Si l'application peut authentifier, elle doit lier cette authentification au "*session ID*", pour être bien sûr qu'un attaquant ne va pas profiter d'une authentification réussie dans une session pour abuser d'une autre. Par exemple, si l'authentification est faite par une méthode analogue à celle du RFC 7616, le "*session ID*" peut être ajouté aux éléments qui seront condensés. Et si la méthode d'authentification ressemble à SCRAM (RFC 5802), le "*session ID*" peut être utilisé comme "*channel binding*".

ENO n'est pas lié à un algorithme cryptographique particulier, en application du principe d'agilité (RFC 7696). Mais cela implique qu'un algorithme faible peut affaiblir la sécurité de tout le système. Les mises en œuvre d'ENO doivent donc faire attention à ne pas accepter des algorithmes cryprographiques faibles.

Pour les mises en œuvre d'ENO, voir la fin de mon article sur le RFC 8548 ; pour l'instant, ce sont les mêmes que celles de tcpcrypt.