RFC 8689 : SMTP Require TLS Option

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 28 novembre 2019

Date de publication du RFC : Novembre 2019

https://www.bortzmeyer.org/8689.html

Ah, la sécurité, c'est toujours compliqué. Pour le courrier électronique, par exemple, SMTP peut être fait sur TLS, pour garantir la confidentialité et l'intégrité du message. Mais TLS est optionnel. Cela entraine deux problèmes : si la politique du MTA est laxiste, le message risque de passer en clair à certains moments, et si elle est stricte, le message risque d'être rejeté alors que l'expéditeur aurait peut-être préféré qu'il passe en clair. Ce nouveau RFC fournit deux mécanismes, un pour exiger du TLS à toutes les étapes, un au contraire pour demander de la bienveillance et de la tolérance et d'accepter de prendre des risques.

SMTP (RFC 5321 [1]) a une option nommée `STARTTLS` (normalisée dans le RFC 3207), qui permet, si le pair en face l'accepte, de passer la session SMTP sur TLS, assurant ainsi sa sécurité. `STARTTLS` a plusieurs problèmes, notamment son caractère optionnel. Même avec des sessions SMTP entièrement TLS (sans `STARTTLS`, cf. RFC 8314), le problème demeure. Que doit faire un MTA s'il ne peut pas lancer TLS, parce que le MTA en face ne l'accepte pas, ou parce que son certificat est invalide (RFC 6125) ou encore car DANE (RFC 7672) est utilisé et que le certificat ne correspond pas ? Jusqu'à présent, la décision était prise par chaque MTA et, comme SMTP repose sur le principe du relayage, l'émetteur original ne pouvait pas exprimer ses préférences, entre « je suis parano, j'ai lu le bouquin de Snowden <https://www.bortzmeyer.org/permanent-record.html>, j'exige du TLS tout le temps » et « je m'en fous, je suis inconscient, je crois que je n'ai rien à cacher <https://www.youtube.com/watch?v=rEwf4sDgxHo>, je veux que le message soit envoyé, même en clair ». La politique des serveurs SMTP était typiquement de privilégier la distribution du message plutôt que sa sécurité. Désormais, il possible pour l'émetteur de donner ses préférences : l'option SMTP `REQUIRETLS` permet d'exiger du TLS tout le temps, et l'en-tête `TLS-Required:` (bien mal nommé) permet d'indiquer qu'on préfère au contraire la délivrance du message à sa sécurité.

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc5321.txt`

En général, aujourd'hui, les MTA acceptent d'établir la session TLS, même si le certificat est invalide. En effet, dans le cas contraire, peu de messages seraient livrés, les certificats dans le monde du courrier étant fréquemment invalides, à l'opposé de ce qui se passe dans le monde du Web, où les navigateurs sont bien plus stricts. Pour durcir cette politique par défaut, et aussi parce qu'un attaquant actif peut retirer l'option `STARTTLS` et donc forcer un passage en clair <`https://moxie.org/software/sslstrip/`>, il existe plusieurs mécanismes permettant de publier une politique, comme DANE (RFC 7672) et MTA-STS (RFC 8461). Mais elles sont contrôlées par le récepteur, et on voudrait permettre à l'émetteur de donner son avis.

Commençons par `REQUIRETLS`, l'extension SMTP. (Désormais dans le registre IANA des extensions SMTP <`https://www.iana.org/assignments/mail-parameters/mail-parameters.xml#mail-parame`
Il s'agit pour l'émetteur d'indiquer qu'il ne veut pas de laxisme : il faut du TLS du début à la fin, et, évidemment, avec uniquement des certificats valides. En utilisant cette extension, l'émetteur indique qu'il préfère que le message ne soit pas distribué, plutôt que de l'être dans de mauvaises conditions de sécurité. Cette extension peut être utilisée entre deux MTA, mais aussi quand un MUA se connecte au premier MTA, pour une soumission de message (RFC 6409). Voici un exemple (« C : » = envoyé par le client, « S : » = envoyé par le serveur) :

```
S: 220 mail.example.net ESMTP
C: EHLO mail.example.org
S: 250-mail.example.net Hello example.org [192.0.2.1]
S: 250-SIZE 52428800
S: 250-8BITMIME
S: 250-REQUIRETLS
C: MAIL FROM:<roger@example.org> REQUIRETLS
S: 250 OK
C: RCPT TO:<editor@example.net>
S: 250 Accepted
C: DATA
S: 354 Enter message, ending with "." on a line by itself

(Le message)
C: .
S: 250 OK
C: QUIT
```

(Le lecteur ou la lectrice astucieux aura remarqué qu'il y a un piège, le risque qu'un attaquant actif ne retire le `REQUIRETLS` du client ou bien du serveur. Ce cas est traité plus loin.)

Dans l'exemple ci-dessus, le serveur a annoncé qu'il savait faire du `REQUIRETLS`, et le client a demandé à ce que l'envoi depuis `roger@example.org` soit protégé systématiquement par TLS. Cela implique que pour toutes les sessions SMTP suivantes :

— L'enregistrement MX soit résolu en utilisant DNSSEC (ou alors on utilise MTA-STS, RFC 8461),
— Le certificat soit valide et soit authentifié par une AC ou par DANE,
— Le serveur suivant doit accepter la consigne REQUIRETLS (on veut une chaîne complète, de l'émetteur au récepteur).

Puisque l'idée est d'avoir du TLS partout, cela veut dire qu'un MTA qui reçoit un message marqué `REQUIRETLS` doit noter cette caractéristique dans sa base et s'en souvenir, puisqu'il devra passer cette exigence au serveur suivant.

Si le serveur en face ne sait pas faire de `REQUIRETLS` (ou, pire, pas de TLS), l'émetteur va créer une erreur commençant par 5.7 (les erreurs SMTP étendues sont décrites dans le RFC 5248) :

```
REQUIRETLS not supported by server: 5.7.30 REQUIRETLS needed
```

Et l'en-tête `TLS-Required:` ? (Ajouté dans le registre IANA des en-têtes <https://www.iana.org/assignments/message-headers/message-headers.xml#perm-headers>.) Il fait l'inverse, il permet à l'émetteur de spécifier qu'il préfère la distribution du message à la sécurité, et qu'il faut donc débrayer les tests qu'on pourrait faire. Ce nom de `TLS-Required:` est mal choisi, car cet en-tête ne peut prendre qu'une seule valeur, `no` (non), comme dans cet exemple amusant du RFC :

```
From: Roger Reporter <roger@example.org>
To: Andy Admin <admin@example.com>
Subject: Certificate problem?
TLS-Required: No
Date: Fri, 18 Jan 2019 10:26:55 -0800

Andy, there seems to be a problem with the TLS certificate
on your mail server. Are you aware of this?

    Roger
```

Si l'en-tête est présent, le serveur doit être plus laxiste que d'habitude et accepter d'envoyer le message même s'il y a des problèmes TLS, même si la politique normale du serveur serait de refuser. Bien sûr, `TLS-Required: no` n'interdit pas d'utiliser TLS, si possible, et l'émetteur doit quand même essayer. Notez aussi que les MTA sont libres de leur politique et qu'on peut parfaitement tomber sur un serveur SMTP qui refuse de tenir compte de cette option, et qui impose TLS avec un certificat correct, même en présence de `TLS-Required: no`.

(Le lecteur ou la lectrice astucieux aura remarqué qu'il y a un piège, le risque qu'un attaquant actif n'ajoute `TLS-Required: no`. Ce cas est traité plus loin.)

Ah, et si on a les deux, `REQUIRETLS` et `TLS-Required: no` ? La section 4.1 du RFC couvre ce cas, en disant que la priorité est à la sécurité (donc, `REQUIRETLS`).

La section 5 de notre RFC couvre le cas des messages d'erreur générés par un MTA lorsqu'il ne peut pas ou ne veut pas envoyer le message au MTA suivant (ou au MDA). Il fabrique alors un message envoyé à l'expéditeur ("*bounce*", en anglais, ou message de non-distribution). Ce message contient en général une bonne partie, voire la totalité du message original. Sa confidentialité est donc aussi importante que celle du message original. Si celui-ci était protégé par `REQUIRETLS`, le courrier d'erreur doit l'être aussi. Le MTA qui génère ce courrier d'erreur doit donc lui-même activer l'extension `REQUIRETLS`. (Notez que, comme le chemin que suivra cet avis de non-remise ne sera pas forcément le même que celui suivi par le message originel, s'il y a un serveur non-`REQUIRETLS` sur le trajet, le courrier d'erreur ne sera pas reçu.)

Si un logiciel ré-émet un message (par exemple un gestionnaire de liste de diffusion transmettant aux membres de la liste, cf. RFC 5598), il devrait, idéalement, appliquer également le `REQUIRETLS` sur le message redistribué. Le RFC ne l'impose pas car, en pratique, cela risquerait d'empêcher la réception du message par beaucoup.

Notre RFC se termine par une longue section 8 sur la sécurité, car les problèmes qu'essaie de résoudre ces solutions sont complexes. Le cas des attaques passives est facile : TLS protège presque parfaitement

contre elles. Mais les attaques actives soulèvent d'autres questions. REQUIRETLS mènera à un refus des connexions SMTP sans TLS, protégeant ainsi contre certaines attaques actives comme le "*SSL stripping*" <https://moxie.org/software/sslstrip/> ou comme une attaque de l'Homme du Milieu avec un mauvais certificat. (Cette dernière attaque est facile aujourd'hui dans le monde du courrier, où bien des serveurs SMTP croient aveuglément tout certificat présenté.) REQUIRETLS protège également contre beaucoup d'attaques via le DNS, en exigeant DNSSEC (ou, sinon, MTA-STS).

Par contre, REQUIRETLS ne protège pas contre un méchant MTA qui prétendrait gérer REQUIRETLS mais en fait l'ignorerait. De toute façon, SMTP sur TLS n'a jamais protégé des MTA intermédiaires, qui ont le texte du message en clair. Si on veut se protéger contre un tel MTA, il faut utiliser PGP (RFC 4880) ou équivalent. (Par contre, le risque de l'ajout d'un TLS-Required: no par un MTA malveillant ne semble pas traité dans le RFC ; PGP ne protège pas contre cela.)

Il peut y avoir un conflit entre TLS-Required: no et la politique du MTA, qui tient absolument à vérifier les certificats des serveurs auxquels il se connecte, via PKIX ou via DANE. Le RFC laisse entendre que le dernier mot devrait revenir à l'expéditeur, au moins si le message a été envoyé via TLS et donc pas modifié en route. (Le cas d'un message reçu en clair - donc pas sécurisé - et demandant de ne pas exiger TLS reste ouvert...)

Et pour finir, l'exemple de session SMTP où le serveur annonçait qu'il gérait REQUIRETLS (en disant 250-REQUIRETLS) était simplifié. Si la session commençait en clair, puis passait à TLS après, avec la commande STARTTLS, le client doit recommencer la session une fois TLS activé, pour être sûr que ce qu'annonce le serveur est réel.

Bien qu'il y ait déjà des programmeurs ayant travaillé sur ce RFC, je ne trouve encore rien du tout dans le source de Postfix, le MTA que j'utilise, même dans la version expérimentale.