RFC 8461 : SMTP MTA Strict Transport Security (MTA-STS)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 9 janvier 2019

Date de publication du RFC : Septembre 2018

https://www.bortzmeyer.org/8461.html

La question de la sécurité du courrier électronique va probablement amener à la publication de nombreux autres RFC dans les années qui viennent...Ce nouveau RFC traite du problème de la sécurisation de SMTP avec TLS; c'est très bien d'avoir SMTP-sur-TLS, comme le font aujourd'hui tous les MTA sérieux. Mais comment le client SMTP qui veut envoyer du courrier va-t-il savoir si le serveur en face gère TLS? Et si on peut compter dessus, refusant de se connecter si TLS a un problème? Notre RFC apporte une solution, publier dans le DNS un enregistrement texte qui va indiquer qu'il faut télécharger (en HTTP!) la politique de sécurité TLS du serveur. En la lisant, le client saura à quoi s'attendre. Ne cherchez pas de déploiement de cette technique sur le serveur qui gère mon courrier électronique, je ne l'ai pas fait, pour les raisons expliquées plus loin.

Cette solution se nomme STS, pour "*Strict Transport Security*". La section 1 du RFC explique le modèle de menace auquel répond cette technique. Sans TLS, une session SMTP peut être facilement écoutée par des espions et, pire, interceptée et modifiée. TLS est donc indispensable pour le courrier, comme pour les autres services Internet. À l'origine (RFC 3207 [1]), la méthode recommandée pour faire du TLS était dite STARTTLS et consistait, pour le MTA contacté, à annoncer sa capacité TLS puis, pour le MTA qui appelle, à démarrer la négociation TLS s'il le souhaitait, et si le serveur en face avait annoncé qu'il savait faire. Une telle méthode est très vulnérable aux attaques par repli, ici connues sous le nom de "*SSL striping*"; un attaquant actif peut retirer l'indication STARTTLS du message de bienvenue du MTA contacté, faisant croire au MTA appelant que TLS ne sera pas possible.

D'autre part, un très grand nombre de serveurs SMTP ont des certificats expirés, ou auto-signés, qui ne permettent pas d'authentifier sérieusement le MTA qu'on appelle. En cas de détournement BGP ou DNS, on peut se retrouver face à un mauvais serveur et comment le savoir, puisque les serveurs authentiques ont souvent un mauvais certificat. Le MTA qui se connecte à un autre MTA est donc confronté

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc3207.txt

à un choix difficile : si le certificat en face est auto-signé, et donc impossible à évaluer, est-ce parce que l'administrateur de ce MTA est négligent, ou bien parce qu'il y a un détournement ? Pour répondre à cette question, il faut un moyen indépendant de connaitre la **politique** du MTA distant. C'est ce que fournit notre RFC (c'est aussi ce que fournit DANE, en mieux).

À noter que ce RFC concerne les communications entre MTA. Pour celles entre MUA et MTA, voyez le RFC 8314, qui utilise un mécanisme assez différent, fondé sur le TLS implicite (au contraire du TLS explicite de STARTSSL).

Donc, la technique décrite dans ce RFC, STS, vise à permettre l'expression d'une politique indiquant :
— Si le receveur a TLS ou pas,
— Ce que devrait faire l'émetteur si TLS (chiffrement ou authentification) échoue.
D'abord, découvrir la politique du serveur distant (section 3 du RFC). Cela se fait en demandant un enregistrement TXT. Si le domaine de courrier est `example.com`, le nom où chercher l'enregistrement TXT est `_mta-sts.example.com`. Regardons chez Gmail :

```
% dig TXT _mta-sts.gmail.com
...
;; ANSWER SECTION:
_mta-sts.gmail.com. 300 IN TXT "v=STSv1; id=20171114T070707;"
```

On peut aussi utiliser le DNS Looking Glass <https://dns.bortzmeyer.org/_mta-sts.gmail.com/TXT>.

Cet enregistrement TXT contient des paires clé/valeur, deux clés sont utilisées :
— `v` qui indique la version de STS utilisée, aujourd'hui `STSv1`,
— et `id` qui indique un numéro de série (ici 20171114T070707) ; si la politique TLS change, on doit penser à modifier ce numéro. (Rappelons que la politique est accessible via HTTP. Dans beaucoup d'organisations, ce n'est pas la même équipe qui gère le contenu DNS et le contenu HTTP, et je prévois donc des difficultés opérationnelles.)
La présence de cet enregistrement va indiquer qu'une politique TLS pour le MTA est disponible, à récupérer via HTTP.

D'autres clés pourront être créées dans le futur, et mises dans le registre IANA <https://www.iana.org/assignments/mta-sts/mta-sts.xml#mta-sts-txt-record-fields>, en suivant la politique « Examen par un expert » du RFC 8126.

Cette politique doit se trouver en (toujours en supposant que le domaine est `example.com`) `https://mta-sts.e` et avoir le type `text/plain`. (Le `.well-known` est décrit dans le RFC 8615.) Le nom `mta-sts.txt` a été ajouté dans le registre IANA <https://www.iana.org/assignments/well-known-uris/well-known-uris.xml>. Essayons de récupérer la politique de Gmail, puisqu'ils ont l'enregistrement TXT :

```
% curl  https://mta-sts.gmail.com/.well-known/mta-sts.txt
version: STSv1
mode: testing
mx: gmail-smtp-in.l.google.com
mx: *.gmail-smtp-in.l.google.com
max_age: 86400
```

On voit que les politiques sont également décrites sous formes de paires clé/valeur, mais avec le deux-points au lieu du égal. On voit aussi que l'utilisation de STS ("*Strict Transport Security*") nécessite un nom spécial (`mta-sts`, qui n'a **pas** de tiret bas car il s'agit d'un nom de machine <https://www.bortzmeyer.org/host-vs-domain.html>) ce qui peut entrer en conflit avec des politiques de nommages locales, et est en général mal perçu. (Cela a fait râler.) L'utilisation des enregistrements de service aurait évité cela mais ils n'ont pas été retenus.

Les clés importantes sont :

— `version` : aujourd'hui `STSv1`,
— `mode` : indique au MTA émetteur ce qu'il doit faire si TLS échoue. Il y a le choix entre `none` (continuer comme si de rien n'était), `testing` (signaler les problèmes - cf. section 6 - mais continuer, c'est le choix de Gmail dans la politique ci-dessus) et `enforce` (refuser d'envoyer le message en cas de problème, c'est le seul mode qui protège contre les attaques par repli).
— `max_age` : durée de vie de la politique en question, on peut la mémoriser pendant ce nombre de secondes (une journée pour Gmail),
— `mx` : les serveurs de messagerie de ce domaine.

D'autres clés pourront être ajoutées dans le même registre IANA <https://www.iana.org/assignments/mta-sts/mta-sts.xml#mta-sts-policy-fields>, avec la politique d'examen par un expert (cf. RFC 8126.) Un exemple de politique figure dans l'annexe A du RFC.

(Notez qu'au début du processus qui a mené à ce RFC, la politique était décrite en JSON. Cela a été abandonné au profit d'un simple format « clé : valeur » car JSON n'est pas habituel dans le monde des MTA, et beaucoup de serveurs de messagerie n'ont pas de bibliothèque JSON incluse.)

La politique **doit** être récupérée en HTTPS (RFC 2818) authentifié (RFC 6125), le serveur doit donc avoir un certificat valide pour `mta-sts.example.com`. L'émetteur doit utiliser SNI ("*Server Name Indication*", RFC 6066), et parler au moins TLS 1.2. Attention, le client HTTPS ne doit pas suivre les redirections, et ne doit pas utiliser les caches Web du RFC 9111.

Notez bien qu'une politique ne s'applique qu'à un domaine, pas à ses sous-domaines. La politique récupérée en `https://mta-sts.example.com/` ne nous dit rien sur les serveurs de messagerie du domaine `something.example.com`.

Une fois que l'émetteur d'un courrier connait la politique du récepteur, il va se connecter en SMTP au MTA du récepteur et vérifier (section 4) que :

— Le MX correspond à un des serveurs listés sous la clé `mx` dans la politique,
— Le serveur accepte de faire du TLS et a un certificat valide. Le certificat ne doit évidemment pas être expiré <https://www.bortzmeyer.org/tester-expiration-certifs.html> (ce qui est pourtant courant) et doit être signé par une des AC connues de l'émetteur (ce qui garantit bien des problèmes, puisque chaque émetteur a sa propre liste d'AC, qui n'est pas connue du récepteur). Le certificat doit avoir un SAN ("*Subject Alternative Name*", cf. RFC 5280) avec un DNS-ID (RFC 6125) correspond au nom du serveur.

L'exigence que le certificat soit signé par une AC connue de l'émetteur est la raison pour laquelle je ne déploie pas STS sur mon domaine `bortzmeyer.org` ; ses serveurs de messagerie utilisent des certificats signés par CAcert <https://www.bortzmeyer.org/cacert.html>, et beaucoup de MTA n'ont pas CAcert dans le magasin qu'ils utilisent. Exiger un certificat signé par une AC connue de l'émetteur pousse à la concentration vers quelques grosses AC, puisque c'est la seule façon d'être raisonnablement sûr qu'elle soit connue partout. Une meilleure solution que STS, et celle que j'utilise, est d'utiliser DANE (RFC 7672). Mais STS est issu de chez Google, qui pousse toujours aux solutions centralisées, avec un petit nombre d'AC officielles. (Cf. annonce initiale <https://security.googleblog.com/2016/03/more-encryption-more-notifications-more.html> où Google dit clairement « nous avons une solution, nous allons la faire normaliser ».)

Et si les vérifications échouent ? Tout dépend de la valeur liée à la clé `mode` dans la politique. Si elle vaut `enforce`, l'émetteur renonce et le message n'est pas envoyé, si le mode est `testing` ou `none`, on envoie quand même. Dans le cas `testing`, l'émetteur tentera de signaler au récepteur le problème (section 6), en utilisant le RFC 8460. C'est donc le récepteur, via la politique qu'il publie, qui décide du sort du message. L'annexe B du RFC donne, en pseudo-code, l'algorithme complet.

La section 8 de notre RFC rassemble quelques considérations pratiques pour celles et ceux qui voudraient déployer cette technique STS. D'abord, le problème du changement de politique. Lorsqu'on modifie sa politique, il y a deux endroits à changer, les données envoyées par le serveur HTTP, et l'enregistrement TXT dans le DNS (pour modifier la valeur associée à `id`). Deux changements veut dire qu'il y a du potentiel pour une incohérence, si on modifie un des endroits mais pas l'autre, d'autant plus que ces deux endroits peuvent être sous la responsabilité d'équipes différentes. Et l'enregistrement TXT est soumis au TTL du DNS. Donc, attention à changer la politique publiée en HTTPS avant de changer l'enregistrement DNS. Et il faut faire en sorte que les deux politiques fonctionnent car certains envoyeurs auront l'ancienne.

Un cas courant en matière de courrier électronique est la délégation de l'envoi du courrier massif à un « routeur » (rien à voir avec les routeurs IP). Ainsi, les "*newsletters*", dont le service marketing croit qu'elles servent à quelque chose, sont souvent déléguées à un spammeur professionnel, le « routeur ». Dans ce cas, il faut aussi leur déléguer la politique, et le `_mta-sts` dans le DNS doit donc être un alias pointant vers un nom du routeur. Idem pour le nom `mta-sts` qui pointe vers le serveur HTTP où récupérer la politique. Cela doit être un alias DNS, ou bien un "*reverse proxy*" HTTP (pas une redirection HTTP, prohibée par le RFC).

Notre RFC se termine par la section 10, qui détaille quelques points de sécurité. Comme toute la sécurité de STS dépend du certificat PKIX qui est présenté à l'émetteur du courrier (cf. RFC 6125), la sécurité de l'écosystème X.509 est cruciale. Si une AC délivre un faux certificat (comme c'est arrivé souvent), l'authentification TLS ne protégera plus.

Autre problème potentiel, le DNS. Sans DNSSEC, il est trop facile de tromper un résolveur DNS et, par exemple, de lui dire que l'enregistrement TXT n'existe pas et qu'il n'y a donc pas de politique STS. La bonne solution est évidemment de déployer DNSSEC, mais le RFC donne également quelques conseils pratiques pour ceux et celles qui s'obstinent à ne pas avoir de DNSSEC. (Celles et ceux qui ont DNSSEC ont de toute façon plutôt intérêt à utiliser DANE.) Notamment, il est recommandé que le serveur émetteur mémorise ce que faisait le récepteur et se méfie si la politique disparait (une forme de TOFU).

STS n'est pas la première technique qui vise à atteindre ces buts. La section 2 de notre RFC en présente d'autres. La principale est évidemment DANE (RFC 7672), celle que j'ai choisi de déployer pour le service de courrier de `bortzmeyer.org`. DANE nécessite DNSSEC, et le RFC présente comme un avantage que la solution qu'il décrit ne nécessite pas DNSSEC (c'est une erreur car, sans DNSSEC, la solution de ce RFC marche mal puisqu'un attaquant pourrait prétendre que l'enregistrement texte n'existe pas, ou bien lui donner une autre valeur). Par contre, il est exact que STS permet un mode « test seulement » que ne permet pas DANE. Mais ce n'est pas un argument suffisant pour moi.

Qui met en œuvre STS aujourd'hui ? On a vu que Gmail le faisait. Yahoo a également une politique STS :

```
% dig +short TXT _mta-sts.yahoo.com
"v=STSv1; id=20161109010200Z;"
```

Elle est en `testing`, comme celle de Gmail (je n'ai encore rencontré aucun domaine qui osait utiliser `enforce`). Comme illustration du fait que la publication de la politique nécessitant **deux** endroits (DNS et HTTP) est casse-gueule, regardons chez Microsoft :

```
% dig +short TXT _mta-sts.outlook.com
"v=STSv1; id=20180321T030303;"
```

Mais en HTTP, on récupère un 404 (ressource non existante) sur la politique…

ProtonMail n'a rien, pour l'instant STS semble uniquement sur les gros silos centralisés :

```
% dig TXT _mta-sts.protonmail.com

; <<>> DiG 9.11.3-1ubuntu1.3-Ubuntu <<>> TXT _mta-sts.protonmail.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 12470
...
```

Et question logiciels libres de MTA, qu'est-ce qui existe ? Voyons Postfix ; il y a deux cas à distinguer :

— Pour le courrier entrant, où Postfix n'est pas concerné, il suffit d'activer TLS sur Postfix <https://www.bortzmeyer.org/postfix-tls.html>, puis de publier sa politique (rappelez-vous : DNS **et** HTTP).

— Pour le courrier sortant, il faudrait que Postfix regarde la politique du serveur distant. Il n'est pas évident que ce code doive être dans le démon Postfix lui-même (trop de code n'est pas une bonne idée). Comme le note Venema, « "*Like DKIM/DMARC I do not think that complex policies like STS should be built into core Postfix SMTP components*" ». Une meilleure solution est sans doute un programme externe comme postfix-mta-sts-resolver <https://pypi.org/project/postfix-mta-sts-resolve>, qui met en œuvre une partie de ce RFC (la possibilité d'envoi de rapports est absente, par exemple), et communique ensuite les résultats à Postfix, via le système "*socketmap*". Vous pouvez lire cet article d'exemple <https://blog.delouw.ch/2018/12/16/using-mta-sts-to-enhance-email-> . Ne me demandez pas des détails, comme expliqué plus haut, j'utilise DANE et pas STS.

Une fois tout cela configuré, il existe un bon outil de test en ligne <https://aykevl.nl/apps/mta-sts/> pour voir si votre politique STS est correcte. Essayez-le avec `outlook.com` pour rire un peu.

Quelques articles sur STS :

— Keltia n'a pas aimé <https://blog.keltia.net/2018/08/23/mta-sts-or-the-worst-of-everything>.

— Shaft non plus <https://mamot.fr/@Shaft/100794747206190506>.

— Le service de tests de sécurité en ligne Hardenize a fait un bon article de synthèse <https://www.hardenize.com/blog/mta-sts>.

— Une autre solution pour connaitre la politique TLS d'un domaine : la liste maintenue par l'EFF <https://starttls-everywhere.org/policy-list/> (je n'y suis pas non plus, puisque l'AC que j'ai choisie n'est pas acceptée par l'EFF).