RFC 7672 : SMTP security via opportunistic DANE TLS

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 18 octobre 2015

Date de publication du RFC : Octobre 2015

https://www.bortzmeyer.org/7672.html

L'état actuel de la sécurisation de SMTP par TLS n'est pas très satisfaisant : la grande majorité des certificats est auto-signée (pour éviter les prix et les délais des Autorités de certification) et, souvent, expirée <https://www.bortzmeyer.org/tester-expiration-certifs.html>. Conséquence logique, les serveurs SMTP ne vérifient pas les certificats : s'ils le faisaient, ils ne pourraient plus envoyer de courrier, ou bien ils se rabattraient sur du SMTP en clair, sans TLS, ce qui serait pire que d'accepter un certificat pourri. Comment améliorer la situation ? Ce nouveau RFC propose d'utiliser DANE en publiant dans le DNS, sécurisé par DNSSEC, le certificat du serveur. DANE était déjà normalisé pour HTTP, voici comment l'utiliser pour le courrier, où il y a un petit piège supplémentaire : le nom du serveur est indiqué via les enregistrements MX.

Le courrier a des spécificités qui font qu'on ne peut pas appliquer directement DANE de la même façon qu'avec HTTP (section 1 du RFC). Outre l'indirection entre nom du service (ce qui est à droite du @ dans une adresse) et nom de serveur (la partie droite de l'enregistrement MX), il y a le fait que rien n'indique si un service ou un serveur donné a TLS. Résultat, un attaquant actif peut supprimer la requête `STARTTLS` (RFC 3207 [1]), forçant ainsi un repli vers l'envoi de courrier en clair (c'est une des plus grosses vulnérabilités de SMTP-sur-TLS).

Généraliser TLS pour SMTP va prendre du temps (en France, il faut apparemment une charte signée avec le ministre <http://proxy-pubminefi.diffusion.finances.gouv.fr/pub/document/18/19874.pdf> juste pour activer TLS sur son Postfix <https://www.bortzmeyer.org/postfix-tls.html>). La sécurité de SMTP va donc forcément être « opportuniste » (RFC 7435), c'est-à-dire « on chiffre quand on peut » (au passage, attention en lisant les textes sur la sécurité en anglais : « *"opportunistic security"* » est un terme flou, qui désigne des choses très différentes selon les auteurs. Ce n'est pas du vocabulaire standardisé de la sécurité).

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

Notre nouveau RFC 7672 propose de résoudre ce problème avec des enregistrements DANE. Cela permet notamment de résister aux attaques par repli telles que celle où un attaquant actif retire `STARTTLS` de la liste des extensions SMTP d'un serveur. De même, DANE permet de se protéger contre un Homme du Milieu, en authentifiant le serveur (ce qui est théoriquement possible avec TLS aujourd'hui mais, comme on l'a vu, n'est quasiment jamais fait). À noter qu'un attaquant actif, avec le courrier d'aujourd'hui, n'a même pas besoin de supprimer `STARTTLS` : s'il n'y a pas de DNSSEC, il peut se contenter d'un empoisonnement DNS pour changer le MX, vers un serveur qui collabore davantage avec lui.

DANE, normalisé dans le RFC 6698, doit sa création à la constatation des faiblesses de X.509. La principale est le nombre excessif d'autorités de certification à qui il faut faire confiance (regardez le magasin de certificats dans votre navigateur Web...) Qu'une seule d'entre elles trahisse <`http://www.01net.com/actualites/comment-le-ministere-des-finances-espionne-le-trafic-web-de-ses-collab` `html`> ou soit piratée et tout est fichu, même pour les gens qui ne sont pas clients de cette AC.

Bon, mais HTTPS utilise TLS et les AC de X.509 depuis des années et, plus ou moins, ça marche. Pourquoi SMTP serait-il différent ? La section 1.3 de notre RFC explique les différences (lisez bien cette partie si vous voulez comprendre en détail le problème qu'on essaie de résoudre) :

— Un URL `https://...` indique clairement la volonté de protéger la session avec TLS. Au contraire, rien dans une adresse de courrier électronique ne permet d'indiquer cette volonté. Cela serait d'ailleurs difficile : SMTP-sur-TLS fonctionne étape par étape alors que l'adresse est de bout en bout. La seule solution est donc d'opportunité : si le serveur SMTP à l'autre bout annonce `STARTTLS` (RFC 3207) dans sa bannière, tenter sa chance en TLS. Cela permet évidemment des attaques par repli (un attaquant actif peut supprimer le `STARTTLS` de la bannière), qui sont la plaie de SMTP-sur-TLS.

— HTTP, hélas, n'utilise pas d'indirection entre le nom de service (par exemple `dnsviz.net`) et un nom de serveur assurant ce service (par exemple `www1.dnsviz.net`). Cela a des tas de conséquences négatives pour HTTP (difficulté de faire de la répartition de charge côté client, obligation de « squatter » l'apex du domaine avec des enregistrements A ou AAAA, etc). Si HTTP était à refaire, il faudrait absolument lui faire utiliser les enregistrements SRV. Mais l'indirection, que ce soit par les enregistrements SRV ou bien par la forme simplifiée, les enregistrements MX, qu'utilise SMTP, a un défaut : il faut sécuriser la liaison entre nom de service (la partie gauche du MX, celle qui apparait dans l'adresse de courrier) et nom de serveur (la partie droite du MX). Imaginons un domaine `internautique.fr` dont le courrier est servi par `mail1.provider.example`. Même si ce dernier utilise TLS, si l'enregistrement MX n'est pas sécurisé (typiquement via DNSSEC), un attaquant pourrait tout simplement rediriger le courrier vers un autre serveur que `mail1.provider.example`, plus favorable à ses noirs desseins. Une solution à ce problème serait que le certificat de `mail1.provider.example` contienne également ("*subject alternative name*" ou un truc de ce genre) le nom du service (ici `internautique.fr`). Ce n'est pas du tout réaliste pour un gros hébergeur ayant des dizaines ou des centaines de milliers de clients.

— Une solution au problème de l'attaque par repli est de configurer en dur les serveurs SMTP pour exiger de certains destinataires une session TLS ou rien du tout. Cela protège bien contre un attaquant actif mais cela ne passe évidemment pas à l'échelle. Une telle méthode est utilisée dans le système Email Made in Germany <`http://www.e-mail-made-in-germany.de/`>, où les gros serveurs SMTP allemands s'engagent à configurer en dur le certificat des autres (peut-être fera-t-on la même chose en France mais cela n'est pas explicite dans les projets actuels). Cela marche au sein d'un cartel de gros, pas pour le courrier en général, où on veut pouvoir écrire à des gens avec qui on n'a pas de relation pré-établie.

— Une des méthodes les plus utilisées par HTTP pour gérer les problèmes de certificat est de demander à l'utilisateur : « le certificat pour `https://www.bortzmeyer.org/` n'est pas signé par une autorité reconnue, voulez-vous continuer ? » D'innombrables études auprès des utilisateurs ont montré que cela ne marchait pas : l'utilisateur ne connait pas assez X.509 et la sécurité pour pouvoir prendre une décision informée. Mais, de toute façon, cette méthode n'a pas de sens pour SMTP, qui n'est pas interactif : le MTA n'a pas d'utilisateur humain à qui demander.

Ça, c'était le problème. Le reste du RFC est consacré à la solution, « DANE pour SMTP ». L'idée est que le MTA émetteur fasse une requête DNS pour le type TLSA (le type utilisé par DANE : ce n'est pas un acronyme, ne cherchez pas sa signification). Une fois les enregistrements TLSA récupérés et validés (par DNSSEC), l'émetteur se connecte au récepteur en SMTP et lance TLS. Il récupère alors le certificat et le compare à ce qu'il a trouvé dans l'enregistrement TLSA. Si c'est bon, il continue, si non il avorte. Un tel système protège contre les attaques par repli : si on trouve un enregistrement TLSA, c'est que le récepteur sait faire du TLS et veut le faire.

Donc, première étape de cette méthode, trouver les TLSA (section 2) du RFC. Si on en trouve, ils indiquent clairement la volonté du pair SMTP de faire du TLS (contrairement à l'indication `STARTTLS` dans la bannière, qui est parfois envoyée par des serveurs qui ne savent pas faire de TLS). C'est donc l'équivalent du HSTS de HTTP (RFC 6797), une promesse. Administrateurs système, ne mettez pas un TLSA dans le DNS si vous ne savez pas ce que vous faites !

En résultat de la requête DNS pour trouver les TLSA, il y a quatre possibilités :

— Une réponse validée avec DNSSEC, et avec au moins un enregistrement utilisable. C'est un engagement du destinataire à faire du TLS. On y va et on chiffre. Si la session TLS échoue ou si le certificat obtenu en TLS ne correspond pas, on doit renoncer, plutôt que de livrer du courrier à l'Homme du Milieu.
— Une réponse validée mais où aucun enregistrement n'est utilisable. Au passage, « utilisable » et « non utilisable » sont définis dans la section 4.1 du RFC 6698, celui qui normalise DANE. Un enregistrement est utilisable s'il est signé, avec signature valide, et si son contenu est accepté par le client (par exemple, le champ « usage du certificat » a une valeur connue). Le client doit alors utiliser TLS mais peut se dispenser d'authentifier, puisqu'il n'a pas d'enregistrement TLSA pour cela.
— Une réponse non validée par DNSSEC ou bien une réponse validée qu'il n'existe aucun enregistrement TLSA. C'est la situation la plus courante actuellement, et on fait comme si DANE n'avait pas été inventé, en tentant TLS, puis en se rabattant sur du texte en clair si TLS échoue.
— Une erreur, notamment un échec DNSSEC, par exemple parce que les données ont été modifiées par un attaquant actif. On ne doit **pas** transmettre le courrier à ce serveur.

En fait, j'ai un peu exagéré, la première étape n'est pas de trouver les enregistrements TLSA mais de trouver les enregistrements MX. Naturellement, on ne doit utiliser DANE **que si** le MX est signé par DNSSEC. S'il y a plusieurs enregistrements MX, certains peuvent pointer vers des serveurs qui n'ont pas DANE, voire pas TLS. Il est important de se rappeler que l'algorithme de choix parmi ces enregistrements MX (RFC 5321, section 5.1) ne tient pas compte de TLS : le premier serveur peut ne pas avoir TLS alors que les suivants l'ont. DANE ne modifie pas les algorithmes de SMTP et le choix d'un relais de courrier ne dépend donc pas de leur sécurité. Si on veut imposer TLS, il faut donc mettre TLS sur tous ses serveurs annoncés par les enregistrements MX. (Si SMTP avait été conçu plus récemment, il utiliserait les enregistrements SRV et on devrait donc lire le RFC 7673 à la place de celui-ci.)

Ah, et où trouver l'enregistrement TLSA ? Le nom de domaine à utiliser avec DANE est `_port._protocole.nom.du.serveur`. Pour SMTP, le port est normalement 25. Donc, en général, on cherchera l'enregistrement TLSA en `_25._tcp....` Voyons un cas réel avec le domaine `sources.org`. On cherche d'abord le MX (je fais ici avec dig ce qu'un MTA DANE ferait automatiquement) :

```
% dig MX sources.org
...
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 5, ADDITIONAL: 1
...
;; ANSWER SECTION:
sources.org. 86400 IN MX 10 uucp.bortzmeyer.org.
```

On trouve un MX, sécurisé par DNSSEC (le "*flag*" AD - "*Authentic Data*" - dans la réponse ; notez au passage qu'il faut un résolveur DNSSEC validant pour faire du DANE). Cherchons maintenant le TLSA :

```
 % dig TLSA _25._tcp.uucp.bortzmeyer.org
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 9, ADDITIONAL: 9
...
;; ANSWER SECTION:
_25._tcp.uucp.bortzmeyer.org. 86400 IN TLSA 2 0 1 (
FF2A65CFF1149C7430101E0F65A07EC19183A3B633EF
4A6510890DAD18316B3A )
```

On a un TLSA utilisable, donc on va faire du TLS et vérifier que le certificat obtenu en TLS correspond bien au contenu de l'enregistrement TLSA.

Et quels enregistrements TLSA suggérer pour les serveurs SMTP, parmi les nombreuses combinaisons possibles ? Notre RFC recommande DANE-EE (usage 3) + SPKI (sélecteur 1) + SHA-256 (méthode de correspondance 1). Pourquoi ces recommandations (vous noterez, sur mon domaine personnel, que je ne les ai pas suivies, utilisant le « second choix » du RFC...) ? DANE-EE (c'est-à-dire qu'on ne fait pas de vérifications X.509 à partir des autorités de certification dans le magasin et que le certificat dans l'enregistrement TLSA est celui du serveur) est recommandé parce que les trois autres possibilités ont des défauts gênants. Avec DANE-TA (usage 2, que j'utilise personnellement), le certificat est celui de l'AC, pas celui du serveur. Comme tous les clients DANE n'ont pas forcément « mon » AC dans leur magasin, il faut envoyer le certificat de l'AC en sus de celui du serveur (ce que je fais sur mon serveur SMTP : cela ne me semble pas si difficile que ça). PKIX-TA et PKIX-EE (où on continue à faire les vérifications X.509 classiques) ont le même problème (la variété des AC présentes dans les différents magasins, qui interdit de compter sur une AC de référence). Mettre ces deux usages est donc dangereux, bien des MTA ne pourront plus vous authentifier.

Pour l'usage DANE-EE, qui est recommandé, le sélecteur suggéré est SPKI (l'enregistrement TLSA contient juste la clé, pas tout le certificat) car les autres informations du certificat (comme la date d'expiration ou le nom du serveur) sont inutiles (toute l'information utile est dans le DNS).

Pour l'usage DANE-TA, en revanche, il est recommandé de mettre un sélecteur CERT, qui indique que le client DANE doit vérifier tout le certificat, pas juste la clé.

Quant aux méthodes de correspondance (le troisième champ de l'enregistrement TLSA), la valeur 0 (on met le certificat entier, au lieu de juste un condensat) est déconseillée, en raison de la difficulté du DNS à faire passer des données de trop grande taille.

Au fait, on a surtout ici parlé de DANE pour sécuriser la communication entre deux serveurs SMTP, deux MTA. Mais entre un MUA et un MTA ? Pour les protocoles comme IMAP (RFC 6186), on utilise DANE+SRV (RFC 7673). Pour SMTP (soumission d'un message par le MUA), qui utilise rarement le RFC 6186 pour l'instant, les configurations statiques traditionnelles marchent encore.

Les administrateurs système seront intéressés par la section 9 du RFC, consacrée aux problèmes opérationnels. Par exemple, un administrateur peut hésiter à activer DANE sur le serveur SMTP dont il a la responsabilité : et si, se dit-il, mon courrier n'est plus transmis, par exemple parce que trop d'informaticiens font des erreurs et publient des configuration DANE cassées ? Cet administrateur prudent peut envisager d'utiliser DANE seulement pour certaines destinations, « pour voir », ou bien utiliser DANE en mode « on regarde et on journalise » mais on ne vérifie pas.

Autre problème opérationnel important, un changement de certificat : il faut bien penser à publier le nouveau TLSA à l'avance, pour que l'ancien ensemble TLSA ait disparu de tous les caches DNS avant que le serveur SMTP n'utilise le nouveau certificat.

Et pour faire du DANE avec SMTP en pratique ? Un MTA qui gère DANE existe, Postfix, depuis la version 2.11 (cf. la documentation officielle <http://www.postfix.org/TLS_README.html#client_tls_dane>). Vous vérifiez bien que votre serveur SMTP utilise un résolveur DNSSEC validant, vous mettez dans la configuration :

```
smtp_dns_support_level = dnssec

smtp_tls_security_level = dane
```

Et c'est parti. Essayons d'envoyer un message à DENIC qui a annoncé avoir déployé DANE <https://www.denic.de/en/denic-in-dialogue/press-releases/press/3947.html> (certificat auto-signé donc normalement pas validable) :

```
Oct 18 21:58:08 aetius postfix/smtp[26542]: Verified TLS connection established \
      to mx2.denic.de[81.91.161.38]:25: \
      TLSv1.2 with cipher DHE-RSA-AES256-GCM-SHA384 (256/256 bits)
```

Et voilà, on se connecte en TLS après avoir vérifié que tout allait bien. En effet, DENIC a un enregistrement TLSA :

```
% dig TLSA _25._tcp.mx1.denic.de
...
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 5
...
;; ANSWER SECTION:
_25._tcp.mx1.denic.de. 3600 IN TLSA 3 0 1 (
17CBD8A164851E86C94A534438D02A4202CC71FCAE6C
24E1F98214BD586F67A1 )
```

Ça, c'était la configuration en émetteur. Et en réception ? Il faut avoir déjà activé TLS sur son serveur <https://www.bortzmeyer.org/postfix-tls.html> (ce qui a été fait il y a pas mal d'années sur tous les serveurs SMTP sérieux). Il faut avoir une zone (correctement) signée avec DNSSEC. Il faut ensuite publier un enregistrement TLSA. Ici, j'ai choisi, comme expliqué plus haut, de publier le certificat de mon AC, CAcert <https://www.bortzmeyer.org/cacert.html>. J'utilise la commande `tlsa` dans l'outil hash-slinger <http://people.redhat.com/pwouters/hash-slinger/> :

```
% tlsa --usage 2 --selector 0 --mtype 1 --output rfc --certificate ~/tmp/cacert.pem \
       --port 25 mail.bortzmeyer.org
_25._tcp.mail.bortzmeyer.org. IN TLSA 2 0 1 ff2a65cff1149c7430101e0f65a07ec19183a3b633ef4a6510890dad18316b3a
```

Et je mets cet enregistrement dans ma zone DNS :

```
% dig TLSA _25._tcp.mail.bortzmeyer.org
...
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 8, ADDITIONAL: 7
...
;; ANSWER SECTION:
_25._tcp.mail.bortzmeyer.org. 86400 IN TLSA 2 0 1 (
FF2A65CFF1149C7430101E0F65A07EC19183A3B633EF
4A6510890DAD18316B3A )
```

Pour tester, on peut faire appel aux copains en leur demandant d'envoyer des messages, ou bien on peut se servir de <https://www.tlsa.info/>, qui teste tout (DNSSEC, DANE et TLS en se connectant au serveur SMTP). Il n'essaie pas uniquement le DNS mais aussi le fait que le certificat corresponde bien à l'enregistrement TLSA. Si je publie un mauvais enregistrement, il proteste :

```
bortzmeyer.fr

The domain lists the following MX entries:
0 uucp.bortzmeyer.org

All TLSA RRs failed. (See details.)

Usable TLSA Records
2, 0, 1 ff2a65cff1149c74[...]10890dad18316b3b - application verification failure: (50)
```

Un autre outil de test est <https://arp.simson.net/dane_check.cgi>.