# RFC 8314 : Cleartext Considered Obsolete: Use of TLS for Email Submission and Access

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 31 janvier 2018

Date de publication du RFC : Janvier 2018

https://www.bortzmeyer.org/8314.html

---

Ce RFC s'inscrit dans le cadre des efforts de sécurisation de l'Internet contre la surveillance massive, efforts notablement accélérés depuis les révélations Snowden. Désormais, l'usage de texte en clair (non chiffré) pour le courrier électronique est officiellement abandonné : POP, IMAP et SMTP doivent être chiffrés.

Plus précisément, ce nouveau RFC du groupe de travail UTA <https://datatracker.ietf.org/wg/uta/about/> vise l'accès au courrier (POP - RFC 1939[1] et IMAP - RFC 3501), ainsi que la **soumission** de messages par SMTP - RFC 6409 (entre le MUA et le premier MTA). Les messages de ces protocoles doivent désormais être systématiquement chiffrés avec TLS - RFC 5246. Le but est d'assurer la confidentialité des échanges. Le cas de la transmission de messages entre MTA est couvert par des RFC précédents, RFC 3207 et RFC 7672 (voir aussi le RFC 8461).

Pour résumer les recommandations concrètes de ce RFC :

— TLS 1.2 au minimum (on croise encore souvent SSL malgré le RFC 7568).
— Migrer le plus vite possible vers le tout-chiffré si ce n'est pas déjà fait.
— Utiliser le « TLS **implicite** » sur un port dédié, où on démarre TLS immédiatement (par opposition au vieux système STARTTLS, où on demandait explicitement le démarrage de la cryptographie, cf. sections 2 et 3 du RFC).

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc1939.txt

Ce RFC ne traite pas le cas du chiffrement de bout en bout, par exemple avec PGP (RFC 4880 et RFC 3156). Ce chiffrement de bout en bout est certainement la meilleure solution mais il est insuffisamment déployé aujourd'hui. En attendant qu'il se généralise, il faut bien faire ce qu'on peut pour protéger les communications. En outre, PGP ne protège pas certaines métadonnées comme les en-têtes (`From:`, `Subject:`, etc), alors qu'un chiffrement TLS du transport le fait. Bref, on a besoin des deux.

La section 3 du RFC rappelle ce qu'est le « TLS implicite », qui est désormais recommandé. Le client TLS se connecte à un serveur TLS sur un port dédié, où tout se fait en TLS, et il démarre la négociation TLS immédiatement. Le TLS implicite s'oppose au « TLS explicite » qui était l'approche initiale pour le courrier. Avec le TLS explicite (RFC 2595 et RFC 3207), le serveur devait annoncer sa capacité à faire du TLS :

```
% telnet smtpagt1.ext.cnamts.fr. smtp
Trying 93.174.145.55...
Connected to smtpagt1.ext.cnamts.fr.
Escape character is '^]'.
220 smtpagt1.ext.cnamts.fr ESMTP CNAMTS (ain1)
EHLO mail.example.com
250-smtpagt1.ext.cnamts.fr Hello mail.example.com [192.0.2.187], pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250 STARTTLS
```

Et le client devait dire qu'il allait démarrer la session TLS avec la commande STARTTLS. L'inconvénient principal de STARTTLS est qu'il est vulnérable à l'attaque « "*SSL stripping*" » où un attaquant actif modifie la communication avant que TLS ne démarre, pour faire croire que le partenaire ne sait pas faire de TLS. (Et il y a aussi la vulnérabilité CERT #555316 <`https://www.kb.cert.org/vuls/id/555316`>.) Bien sûr, les serveurs peuvent se protéger en refusant les connexions sans STARTTLS mais peu le font. L'approche STARTTLS était conçue pour gérer un monde où certains logiciels savaient faire du TLS et d'autres pas, mais, à l'heure où la recommandation est de faire du TLS **systématiquement**, elle n'a plus guère d'utilité. (La question est discutée plus en détail dans l'annexe A. Notez qu'un des auteurs de notre nouveau RFC, Chris Newman, était l'un des auteurs du RFC 2595, qui introduisait l'idée de STARTTLS.)

Avec le TLS implicite, cela donne :

```
% openssl s_client -connect mail.example.com:465
...
subject=/CN=mail.example.com
issuer=/O=CAcert Inc./OU=http://www.CAcert.org/CN=CAcert Class 3 Root
...
Peer signing digest: SHA512
Server Temp Key: ECDH, P-256, 256 bits
...
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
...
220 mail.example.com ESMTP Postfix
EHLO toto
250-mail.example.com
250-AUTH DIGEST-MD5 NTLM CRAM-MD5
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
250 SMTPUTF8
```

Donc, concrètement, pour POP, cela veut dire établir la connexion sur le port du TLS implicite, le 995 (et non pas sur le port 110, prévu pour le texte en clair), lancer TLS et authentifier avec le RFC 7817. Puis on fait du POP classique. Pour IMAP, c'est le port 993. Dans les deux cas, cette recommandation de notre RFC ne sera pas trop dure à suivre, le TLS implicite est déjà courant pour ces deux protocoles.

Pour la soumission SMTP (RFC 6409), c'est le port 465 (utilisé auparavant pour du SMTP classique, non-soumission, cf. le registre IANA <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml> et les section 7.3 et annexe A du RFC qui justifient cette réaffectation). Le mécanisme avec STARTTLS sur le port 587 (TLS explicite) est très répandu, contrairement à ce qui se passe pour POP et IMAP. La transition sera donc plus longue, et il faudra donc maintenir les deux services, TLS implicite et explicite, pendant un certain temps.

Voici les serveurs IMAP (pour récupérer le courrier) et SMTP (pour soumettre du courrier) conformes aux recommandations de ce RFC (TLS implicite), tels qu'affichés par Thunderbird :

Et voici la configuration du MTA Postfix pour accepter du TLS implicite sur le port 465 :

```
submissions inet n       -       -       -       -       smtpd
   -o syslog_name=postfix/submissions
   -o smtpd_tls_wrappermode=yes
   -o smtpd_sasl_auth_enable=yes
   -o smtpd_client_restrictions=permit_sasl_authenticated,reject
   -o smtpd_etrn_restrictions=reject
   -o smtpd_sasl_authenticated_header=yes
```

(À mettre dans le `master.cf`.) Cette configuration active TLS et exige une authentification du client (ce qui est normal pour soumettre du courrier.) Pensez aussi à vérifier que le port est bien défini dans `/etc/services` (`smtps 465/tcp ssmtp submissions`).

La section 4 du RFC fournit des détails sur l'utilisation de TLS, du côté des fournisseurs de service de courrier. Le point essentiel est « chiffrement partout, tout le temps » (section 4.1). L'objectif ne pourra pas forcément être atteint immédiatement par tout le monde mais il faut commencer. Par exemple, on peut interdire tout accès en texte clair à certains utilisateurs, puis généraliser à tous. Et, dans ce cas, le message envoyé doit être indépendant de si le mot de passe était valide ou pas (pour ne pas donner d'indication à un éventuel écoutant). Le RFC conseille aussi de traiter les accès avec les vieilles versions de TLS (ou, pire, avec SSL) de la même façon que les accès en clair. Une fois l'accès en clair coupé, il est recommandé de forcer un changement de mot de passe, si l'ancien avait été utilisé en clair.

Et les autres points ? Les fournisseurs de services de courrier électronique doivent annoncer les services POP, IMAP et soumission SMTP en utilisant les enregistrements SRV du RFC 6186, afin de faciliter la tâche des MUA. Un nouvel enregistrement SRV arrive avec ce RFC, d'ailleurs, le `_submissions._tcp`, pour la soumission SMTP sur TLS. Le RFC demande que ces enregistrements SRV (ainsi que les enregistrements MX utilisés pour le courrier entrant, donc a priori pas le sujet de ce RFC) soient signés avec DNSSEC. Et, à propos du DNS, le RFC demande également la publication d'enregistrements TLSA (RFC 6698).

La cryptographie, c'est bien, mais il est également souhaitable de signaler qu'elle a été utilisée, et dans quelles conditions, pour informer les utilisateurs de la sécurité dont ils ont pu bénéficier. Par exemple, le RFC demande que les algorithmes de cryptographie utilisées soient mis dans l'en-tête `Received:` du courrier (cf. aussi le RFC 3848), via une clause `tls` (dans registre IANA <https://www.iana.org/assignments/mail-parameters/mail-parameters.xml#mail-parameters-8>). Notez que beaucoup de serveurs SMTP le font déjà, avec une syntaxe différente :

```
Received: from mail4.protonmail.ch (mail4.protonmail.ch [185.70.40.27])
        (using TLSv1.2 with cipher ECDHE-RSA-AES256-GCM-SHA384 (256/256 bits))
        (No client certificate requested)
        by mail.bortzmeyer.org (Postfix) with ESMTPS id 38DAF31D16
        for <stephane@bortzmeyer.org>; Sat, 20 Jan 2018 16:56:59 +0100 (CET)
```

La section 5 décrit les exigences pour l'autre côté, le client, cette fois, et non plus le serveur de la section 4. D'abord, comme les MUA sont en contact direct avec l'utilisateur humain, ils doivent lui indiquer clairement le niveau de confidentialité qui a été obtenu (par exemple par une jolie icône, différente si on a utilisé TLS ou pas). Notez que cette indication de la confidentialité est un des points du projet Caliopen <https://www.caliopen.org>. Gmail, par exemple, peut afficher ces informations <https://support.google.com/mail/answer/6330403> et cela donne :

Comment est-ce qu'un MUA doit trouver s[Caractère Unicode non montré [2] ]on[Caractère Unicode non montré ]les serveur[Caractère Unicode non montré ]s, au fait ? La méthode recommandée est d'utiliser les enregistements SRV du RFC 6186. Aux _imap._tcp et _pop3._tcp du RFC précédent s'ajoute _submissions._tcp pour le SMTP de soumission d'un message, avec TLS implicite. Attention, le but étant la confidentialité, le MUA ne devrait pas utiliser les serveurs annoncés via les SRV s'ils ne satisfont pas à des exigences minimales de sécurité TLS. Le MUA raisonnable devrait vérifier que le SRV est signé avec DNSSEC, ou, sinon, que le serveur indiqué est dans le même domaine que le domaine de l'adresse de courrier. La section 5.2 donne d'autres idées de choses à vérifier (validation du certificat du serveur suivant le RFC 7817, de la version de TLS…) D'une manière générale, le MUA ne doit pas envoyer d'informations sensibles comme un mot de passe si la session n'est pas sûre. La tentation pourrait être d'afficher à l'utilisateur un dialogue du genre « Nous n'avons pas réussi à établir une connexion TLS satisfaisante car la version de TLS n'est que la 1.1 et le groupe Diffie-Hellman <https://www.bortzmeyer.org/logjam-thunderbird.html> ne fait que 512 bits, voulez-vous continuer quand même ? » Mais le RFC s'oppose à cette approche, faisant remarquer qu'il est alors trop facile pour l'utilisateur de cliquer OK et de prendre ainsi des risques qu'il ne maitrise pas.

Autre question de sécurité délicate, l'épinglage du certificat. Il s'agit de garder un certificat qui a marché autrefois, même s'il ne marche plus, par exemple parce qu'expiré. (Ce n'est donc pas le même épinglage que celui qui est proposé pour HTTP par le RFC 7469.) Le RFC autorise cette pratique, ce qui est du bon sens : un certificat expiré, ce n'est pas la même chose qu'un certificat faux. Et ces certificats expirés sont fréquents car, hélas, bien des administrateurs système ne supervisent pas l'expiration des certificats <https://www.bortzmeyer.org/tester-expiration-certifs.html>. Voici la configuration d'Icinga pour superviser un service de soumission via SMTP :

```
apply Service "submissions" {
  import "generic-service"

  check_command = "ssmtp"
  vars.ssmtp_port = 465
  assign where (host.address || host.address6) && host.vars.submissions
  vars.ssmtp_certificate_age = "7,3"
}
```

2. Car trop difficile à faire afficher par LaTeX

Et, une fois ce service défini, on peut ajouter à la définition d'un serveur `vars.submissions = true` et il sera alors supervisé :

Notre RFC recommande également aux auteurs de MUA de faire en sorte que les utilisateurs soient informés du niveau de sécurité de la communication entre le client et le serveur. Tâche délicate, comme souvent quand on veut communiquer avec les humains. Il ne faut pas faire de fausses promesses (≪ votre connection est cryptée avec des techniques military-grade, vous êtes en parfaite sécurité ≫) tout en donnant quand même des informations, en insistant non pas sur la technique (≪ votre connexion utilise ECDHE-RSA-AES256-GCM-SHA384, je vous mets un A+ ≫) mais sur les conséquences (≪ Ce serveur de courrier ne propose pas de chiffrement de la communication, des dizaines d'employés de la NSA, de votre FAI, et de la Fsociety sont en train de lire le message où vous parlez de ce que vous avez fait hier soir. ≫).

Voilà, vous avez l'essentiel de ce RFC. Quelques détails, maintenant. D'abord, l'interaction de ces règles avec les antivirus et antispam. Il y a plusieurs façons de connecter un serveur de messagerie à un logiciel antivirus et[Caractère Unicode non montré ]ou antispam (par exemple l'interface Milter, très répandue). Parfois, c'est via SMTP, avec l'antivirus et[Caractère Unicode non montré ]ou antispam qui se place sur le trajet, intercepte les messages et les analyse avant de les retransmettre. C'est en général une mauvaise idée (RFC 2979). Dès qu'il y a du TLS dans la communication, c'est encore pire. Puisque le but de TLS est de garantir l'authenticité et l'intégrité de la communication, tout ≪ intercepteur ≫ va forcément être très sale. (Et les logiciels qui font cela sont d'abominables daubes <https://www.bortzmeyer.org/killed-by-proxy.html>.)

Ah, et un avertissement lié à la vie privée (section 8 du RFC) : si on présente un certificat client, on révèle son identité à tout écoutant. Le futur TLS 1.3 aidera peut-être à limiter ce risque mais, pour l'instant, attention à l'authentification par certificat client.

Si vous aimez connaitre les raisons d'un choix technique, l'annexe A couvre en détail les avantages et inconvénients respectifs du TLS implicite sur un port séparé et du TLS explicite sur le port habituel. La section 7 du RFC 2595 donnait des arguments en faveur du TLS explicite. Certaines des critiques que ce vieux RFC exprimait contre le TLS implicite n'ont plus de sens aujourd'hui (le RFC 6186 permet que le choix soit désormais fait par le logiciel et plus par l'utilisateur, et les algorithmes export sont normalement abandonnés). D'autres critiques ont toujours été mauvaises (par exemple, celle qui dit que le choix entre TLS et pas de TLS est binaire : un MUA peut essayer les deux.) Outre la sécurité, un avantage du port séparé pour TLS est qu'il permet de déployer des frontaux - répartiteurs de charge, par exemple - TLS génériques.

Et pour terminer, d'autres exemples de configuration. D'abord avec mutt, par Keltounet <https://mastodon.social/@Keltounet> :

```
smtp_url=smtps://USER@SERVER:465
```

Et le courrier est bien envoyé en TLS au port de soumission à TLS implicite.

Avec Dovecot, pour indiquer les protocoles utilisés dans la session TLS, Shaft <https://mamot.fr/@Shaft/> suggère, dans `conf.d/10-logging.conf` :

```
login_log_format_elements = user=<%u> method=%m rip=%r lip=%l mpid=%e %c %k
```

(Notez le `%k` à la fin, qui n'est pas dans le fichier de configuration d'exemple.)

Et pour OpenSMTPd ? n05f3ra1u <https://mamot.fr/@n05f3ra1u> propose :

```
pki smtp.monserveur.example key "/etc/letsencrypt/live/monserveur.example/privkey.pem"
pki smtp.monserveur.example certificate "/etc/letsencrypt/live/monserveur.example/cert.pem"
...
listen on eno1 port 465 hostname smtp.monserveur.example smtps pki smtp.monserveur.example auth mask-source
```