RFC 6066 : Transport Layer Security (TLS) Extensions: Extension Definitions

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 18 janvier 2011

Date de publication du RFC : Janvier 2011

http://www.bortzmeyer.org/6066.html

Ce RFC décrit les extensions au protocole cryptographique TLS (résumées en section 1.1). Je n'en cite que certaines ici, lisez le RFC pour une liste complète. Ces extensions ne changent pas le protocole : un client ou un serveur TLS qui ne les comprend pas pourra toujours interagir avec ceux qui les comprennent. Le serveur doit en effet ignorer les informations contenues dans le `Hello Client` qu'il ne connait pas (section 7.4.1.2 du RFC 5246 [1]).

Le mécanisme général des extensions figure dans le RFC 5246, notamment la section 7 de ce dernier (voir la structure en 7.4.1.2). En gros, le principe est d'ajouter des données à la fin du paquet `Hello`, qui marque le début de la négociation TLS. Dans le langage de spécification propre à TLS, la liste des extensions possibles est un `enum` (indiqué en section 1.1) mais elle est désormais en ligne à l'IANA <https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xhtml>.

SNI ("*Server Name Indication*", section 3) permet au client TLS d'indiquer au serveur le nom sous lequel il l'a trouvé. Cela autorise le serveur à présenter des certificats différents selon le nom sous lequel on y accède, fournissant ainsi une solution au problème récurrent de l'authentification avec TLS lorsqu'un serveur a plusieurs noms <http://www.bortzmeyer.org/auth-x509-plusieurs-noms.html>. À noter que SNI fonctionne avec les IDN, le RFC décrivant le mécanisme à suivre. Si le RFC 4366 permettait au client d'envoyer plusieurs noms, cette possibilité a été supprimée.

"*Maximum Fragment Length*", section 4, permet au client d'indiquer qu'il souhaiterait des fragments de données de taille plus réduite. Cela peut être utile pour un client contraint en mémoire ou en capacité réseau, par exemple un téléphone portable. Certains de ces clients contraints apprécieront également

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc5246.txt`

"*Truncated HMAC*" (section 7) qui autorise à réduire la taille du MAC utilisé pour protéger l'intégrité des paquets.

Normalement, dans TLS, le certificat du client est envoyé pendant la session TLS. Mais l'extension "*Client Certificate URLs*" (section 5) permet au client d'indiquer un URL où le serveur ira chercher le certificat, allégeant ainsi le travail du client (là encore, c'est prévu pour les clients limités en puissance ou en capacité réseau). Une somme de contrôle en SHA-1 est ajoutée à la liste des URLs envoyés, pour permettre davantage de vérifications (autrement, le serveur risquerait, en cas d'empoisonnement DNS, par exemple, de récupérer un mauvais certificat). Dans le même esprit, la section 8 décrit ("*Certificate Status Request*"), une extension qui permet d'indiquer la volonté d'utiliser OSCP (RFC 6960), un protocole d'interrogation de la validité de certificats X.509 (plus léger que l'habituel téléchargement de la liste de certificats révoqués, suivi d'un traitement local).

Qui dit nouvelles extensions, dit nouvelles erreurs. La section 9 est donc consacrée aux nouveaux codes d'erreur, pour toutes les extensions décrites. La plus évidente étant `unsupported_extension` où le client TLS reçoit une extension qu'il n'avait pas demandé (mise dans son `Hello`). Certaines n'existaient pas dans le RFC 4366 comme `certificate_unobtainable`, utilisé si le mécanisme de "*Client Certificate URLs*" de la section 5 n'arrive pas à récupérer le certificat indiqué.

Tout le protocole TLS servant à la sécurité, il est normal de se demander si les extensions en question vont avoir un effet positif ou pas sur la sécurité. C'est ce que fait la section 11, qui examine les points forts ou faibles de la sécurité de chaque extension. SNI ne semble pas poser trop de problème, par contre "*Client Certificate URLs*" fait que le serveur TLS va jouer un rôle de client (en général, avec le protocole HTTP), ce qui soulève des questions : le client TLS ne va t-il pas utiliser cette extension pour pousser le serveur à attaquer un serveur HTTP, d'autant plus que le client TLS contrôle complètement l'URL ? Ou bien le client TLS coincé derrière un pare-feu ne va-t-il pas essayer de convaincre le serveur TLS de récupérer quelque chose de son côté du pare-feu ? Un serveur HTTP normalement non accessible depuis l'Internet pourrait ainsi être indirectement attaqué, si un serveur TLS gérant cette extension se trouve sur son réseau. Il est donc recommandé que cette extension ne soit pas activée par défaut et seul un jeu limité de plans d'URL soit accepté (ce dernier conseil me semble vain : le plan `http:` sera forcément sur la liste et c'est sans doute le plus intéressant pour un attaquant).

Quant à "*Truncated HMAC*", elle peut aboutir à affaiblir la sécurité puisque le HMAC court est forcément moins fiable. Ce n'est pas forcément très grave car une attaque par force brute contre ce HMAC ne peut pas être faite hors-ligne mais doit être exécutée pendant la session TLS et qu'une seule erreur de la part de l'attaquant coupe la session.

L'annexe A résume les changements depuis la RFC 4366. Le principal est que la RFC 4366 spécifiait à la fois un mécanisme d'extension et des extensions spécifiques. Le mécanisme d'extension ayant été mis dans le RFC 5246 (section 7.4.1.2 à 7.4.1.4), notre RFC 6066 ne contient plus que les extensions.

D'autre part, l'extension SNI a été simplifié en éliminant la possibilité de noms en UTF-8. Désormais, IDN ou pas, le nom doit être représenté en ASCII. Et le condensat cryptographique sur les certificats reçus par l'extension "*Client Certificate URLs*" est désormais obligatoire.