Quelques pensées sur la faille de renégociation de TLS

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 18 novembre 2009. Dernière mise à jour le 25 novembre 2009

http://www.bortzmeyer.org/tls-renego.html

Beaucoup d'électrons ont déjà été secoués pour communiquer des informations sur la faille de sécurité de **renégociation** de TLS. Je ne vais pas martyriser davantage ces pauvres leptons, uniquement faire partager quelques réflexions que m'a inspiré cette faille.

J'ai d'autant moins l'intention d'expliquer en détail la faille que je ne suis pas un expert TLS. D'accord, j'ai lu la norme, le RFC 5246 [1], mais cela ne suffit pas. Et, de toute façon, les articles de Marsh Ray, le découvreur (« "*Renegotiating TLS*" <http://extendedsubset.com/?p=8> » ou d'Eric Rescorla, le gourou TLS de l'IETF (« "*Understanding the TLS renegotiation attack*" <http://www.educatedguesswork.org/2009/11/understanding_the_tls_renegoti.html> ») sont très clairs.

Non, je voudrais plutôt gloser sur quelques points liés à cette faille. D'abord, il faut se rappeler qu'il s'agit d'une faille du protocole, pas d'une mise en œuvre particulière. Les protocoles, comme les programmes, ont des bogues. L'utilisation (parfois) de méthodes formelles pour les spécifier n'a pas diminué le nombre de bogues. (Après tout, les programmes sont tous écrits dans un langage formel et ont quand même plein de bogues.) Néanmoins, certains choix d'implémentation peuvent aggraver la faille. Ainsi, OpenSSL, par défaut, permet la renégociation, même si l'application qui utilise cette bibliothèque n'a rien demandé. GnuTLS, au contraire, ne la permet que si l'application l'a explicitement demandée, ce qui rend ses utilisateurs moins vulnérables.

Ensuite, pourquoi y a t-il des failles dans le protocole TLS ? Une des raisons est sa complexité, le pire ennemi de la sécurité. La norme (RFC 5246) fait plus de cent pages et offre plein d'options et de choix, dont cette fameuse renégociation. Simplifier la norme, retirer des options non indispensables comme la renégociation, aurait probablement diminué le nombre de failles dans TLS.

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

Évidemment, dans le monde réel, les choses ne sont jamais aussi simples. Un protocole offrant moins de possibilités aurait peut-être eu moins de succès. Par exemple, avec HTTPS, si on n'a pas de possibilité de renégociation, la seule façon d'authentifier un client avec un certificat est d'exiger systématiquement le certificat au début de la session TLS. Il ne serait plus possible d'attendre la requête du client, puis de renégocier si cette requête s'avère nécessiter une authentification. Avoir deux serveurs, `www.example.org` et `auth.example.org` est bien sûr possible, mais pas forcément pratique (rendre public un document signifierait un changement d'URL, ce qui est presque toujours une mauvaise idée <http://www.w3.org/Provider/Style/URI>.

En relisant la liste des failles de sécurité qui ont affecté TLS et son prédécesseur SSL, on voit que la faille de renégociation n'est pas la première à reposer sur un problème de liaison ("*binding*") entre deux élements du protocole. Les oublis de liaison sont une des erreurs les plus fréquentes dans les protocoles de sécurité. Supposons un protocole au dessus de TCP qui authentifie, mettons par cryptographie, au début de la connexion. On se croit en sécurité par la suite ? Erreur. Si les paquets TCP suivant l'authentification ne sont pas **liés** à la session qui a été authentifiée (par exemple par une MAC), l'attaquant pourra attendre l'authentification, puis injecter des paquets TCP mensongers qui seront acceptés. De même, dans la faille de renégociation, il n'y avait pas de liaison entre l'« ancienne » session (avant la renégociation) et la « nouvelle », ce qui permettait l'injection de trafic.

Ce qui nous amène aux solutions. Le correctif <http://www.openssl.org/news/secadv_20091111.txt> apporté par OpenSSL supprime simplement la renégociation (ce qui va dans la sens de mes remarques sur le danger des protocoles trop riches en fonctions). Mais cela va empêcher certaines applications de fonctionner. C'est donc un contournement, avec effets de bord désagréables, pas une vraie solution.

Une solution à plus long terme a été élaborée à l'IETF dans le groupe de travail TLS <http://tools.ietf.org/wg/tls>. Elle est décrite dans le RFC 5746. Son principe est de créer une liaison entre l'« ancienne » session et la nouvelle, par le biais d'une nouvelle extension TLS, nommée `renegotiation_info`. Le problème est que les clients TLS ne peuvent pas l'utiliser contre les anciens serveurs (qui ne connaissent pas cette extension) et que le déploiement risque donc d'être laborieux. Pendant un certain temps, les clients TLS auront donc le choix entre exiger cette extension (au risque de ne pas pouvoir se connecter) ou bien accepter le risque de tomber sur un serveur vulnérable à la faille de renégociation. (Il existe un projet concurrent, `draft-mrex-tls-secure-renegotiation`, qui inclus une bonne description de la faille.)

Et pour finir sur une note pratique, un bon moyen (merci à Kim-Minh Kaplan) pour tester si un serveur permet la renégociation (et est donc peut-être vulnérable) :

```
% openssl s_client -connect www.example.org:443
CONNECTED(00000003)
...
R        <--- Tapez cette unique lettre pour commencer la renégociation
RENEGOTIATING
depth=2 /C=ES/ST=BARCELONA/L=BARCELONA/O=IPS Seguridad CA/OU=Certificaciones/CN=IPS SERVIDORES/emailAddress=
verify error:num=19:self signed certificate in certificate chain
verify return:0
```

Ici, le serveur était vulnérable, il a accepté la renégociation. Sinon, il aurait juste affiché :

```
RENEGOTIATING

[Puis plus rien]
```

Une exploitation publique de la vulnérabilité existe, en <http://www.redteam-pentesting.de/en/publications/tls-renegotiation/-tls-renegotiation-vulnerability-proof-of-concept