RFC 7568 : Deprecating Secure Sockets Layer Version 3.0

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 26 juin 2015

Date de publication du RFC : Juin 2015

https://www.bortzmeyer.org/7568.html

Ce nouveau RFC formalise un point déjà bien connu des experts en sécurité : le dernier survivant de la famille SSL, SSL version 3, a de graves failles et ne **doit pas** être utilisé.

Il y a bien longtemps que SSL est dépassé, remplacé par TLS (même si des ignorants continuent de parler de SSL dès qu'ils voient un petit cadenas sur leur navigateur Web). Mais les vieux protocoles ne meurent pas facilement sur l'Internet. Manque de ressources humaines qualifiées, absence d'intérêt pour la sécurité, blocage par la direction financière qui demande quel est le ROI, des tas de raisons font que pas mal de serveurs acceptent encore le SSL v3. Celui-ci, décrit dans le RFC 6101 [1] (mais qui avait été défini et déployé bien avant, en 1996, le RFC 6101 ne faisant que documenter longtemps après), devrait pourtant être abandonné définitivement, ne laissant que TLS (dont la version actuelle est la 1.2, en RFC 5246).

La première fin de SSL v3 était en 1999, avec la sortie de TLS, dans le RFC 2246. Le fait que SSL v3 soit toujours bien vivant aujourd'hui, plus de quinze ans après, donne une idée de l'ossification de l'Internet et du manque de réactivité des techniciens.

La section 3 de notre RFC résume la bonne pratique actuelle : **NE PAS UTILISER SSL v3**. Les clients ne doivent pas le proposer dans le `ClientHello`, ni les serveurs dans leur `ServerHello`.

Bien plus longue, la section 4 décrit pourquoi SSL v3 est cassé sans espoir de réparation. Première raison, qui a permis la faille POODLE (cf. un bon article d'explication pour tous <http://www.slate.fr/story/93437/poodle-bug-caniche-peut-vous-devoiler>), c'est que le remplissage en mode CBC n'est pas spécifié (et donc pas vérifiable par le pair). Autre faille, SSL v3 fait forcément du

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc6101.txt`

"*MAC-then-encrypt*", ce qui est aujourd'hui considéré comme dangereux. TLS fait aussi du "*MAC-then-encrypt*" par défaut mais on peut le changer (RFC 7366) grâce aux extensions. Or, SSL v3 n'a pas la notion d'extensions.

Les algorithmes de chiffrement "*stream*" de SSL v3 ont tous des défauts sérieux, par exemple RC4 est désormais officiellement abandonné (RFC 7465). Question condensation, ce n'est pas mieux, certaines opérations de SSL v3 imposant MD5 (RFC 1321, abandonné par le RFC 6151).

L'échange de clés dans SSL v3 a aussi des failles permettant à un homme du milieu de se placer. Corrigées dans TLS par des extensions, elles ne sont pas corrigeables en SSL v3.

Toujours en raison de son manque d'extensibilité, SSL v3 ne peut pas avoir certaines des nouvelles fonctions développées depuis sa naissance. Pas d'algorithmes dans les modes AEAD (RFC 5246, section 6), pas de courbes elliptiques (RFC 8422), pas de reprise d'une session existante sans état sur le serveur (RFC 5077), pas de mode datagramme pour UDP (RFC 6347), pas de négociation de l'application utilisée (RFC 7301).

Comme le notait le texte d'accompagnement à l'IESG, où on doit indiquer les mises en œuvre du protocole, « "*Are there existing implementations of the protocol? Yes, and that's the problem ;-)*" ». Ceci dit, une bonne partie des déploiements de TLS ont commencé à supprimer SSL v3. Logiquement, il devrait être retiré du code des bibliothèques TLS (alors que, encore aujourd'hui, la page d'accueil de GnuTLS se vante de gérer SSL v3). Espérons que ce RFC, qui n'apporte aucune information technique nouvelle, par son côté « officiel », contribue à faire bouger les choses.

Pour déterminer si un serveur accepte SSL v3, on peut utiliser le client d'OpenSSL :

```
% openssl s_client -connect www.bortzmeyer.org:443 -ssl3
CONNECTED(00000003)
depth=0 CN = www.bortzmeyer.org
...
SSL handshake has read 2276 bytes and written 470 bytes
...
SSL-Session:
    Protocol  : SSLv3
    Cipher    : DHE-RSA-AES256-SHA
    Session-ID: 8B815B79611AB6FB105E84AA3159A0E57D50BA61BEE33478BB5DFEAFD5E4979B
```

Ici, le serveur a accepté SSL v3 ("*handshake*" terminé, algorithme de chiffrement - `DHE-RSA-AES256-SHA` - sélectionné alors qu'on a forcé SSL v3). Si vous voulez savoir pourquoi ce serveur particulier accepte malheureusement SSL v3, voir à la fin de cet article.

Avec un qui n'a pas SSL v3, on voit :

```
% openssl s_client -connect www.example.net:443 -ssl3
CONNECTED(00000003)
3073382076:error:1408F10B:SSL routines:SSL3_GET_RECORD:wrong version number:s3_pkt.c:348:
---
no peer certificate available
---
No client certificate CA names sent
---
SSL handshake has read 5 bytes and written 7 bytes
---
...
SSL-Session:
    Protocol  : SSLv3
    Cipher    : 0000
    Session-ID:
```

Les messages d'OpenSSL, comme toujours, sont peu clairs, mais on peut voir que le "*handshake*" n'a pas abouti (très peu de données transmises), qu'aucun algorithme de chiffrement n'a été choisi et qu'aucune session n'a commencé (ID vide). Ce serveur est donc correct.

Notez que l'administrateur système est dépendant des logiciels utilisés et qu'il ne peut pas tout reprogrammer lui-même. Ainsi, GnuTLS a un module Apache, mod_gnutls `<https://mod.gnutls.org/>`, qui a une très sérieuse bogue `<https://mod.gnutls.org/ticket/29>` (ancienne et jamais réparée) qui fait que SSLv3 n'est pas interdit, même quand la configuration le demande. Personnellement, je vais arrêter d'utiliser GnuTLS sur mes sites Web, pas à cause de la bibliothèque elle-même, qui est excellente, mais à cause de ce module Apache pas maintenu.