RFC 8492 : Secure Password Ciphersuites for Transport Layer Security (TLS)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 22 février 2019

Date de publication du RFC : Février 2019

https://www.bortzmeyer.org/8492.html

Ce nouveau RFC décrit un mécanisme permettant une authentification lors de l'utilisation de TLS sans utiliser de cryptographie à clé publique, mais avec un mot de passe.

Au passage, l'auteur réhabilite la notion de mot de passe, souvent considérée comme une méthode d'authentification dépassée. Il estime que « un mot de passe est plus naturel qu'un certificat » car « depuis l'enfance, nous avons appris la sémantique d'un secret partagé ».

Experts en sécurité, ne jetez pas tout de suite ce RFC à la poubelle. Rassurez-vous, le mot de passe n'est pas utilisé tel quel, mais via un protocole nommé TLS-PWD, dont la principale partie s'appelle Dragonfly. Dragonfly était déjà utilisé dans le RFC 5931 [1].

D'abord, pourquoi ce nouveau mécanisme (section 1 du RFC) ? TLS (RFC 5246) utilise traditionnellement de la cryptographie à clé publique pour l'authentification. La machine qui veut être authentifiée (en général, c'est le serveur, mais le client peut le faire aussi) présente un certificat, contenant la clé publique et diverses métadonnées plus ou moins utiles (date d'expiration, signature d'une AC). Le protocole vérifie ensuite que la machine peut signer des données avec la clé privée correspondante, prouvant ainsi qu'elle connait cette clé privée. Ce mécanisme est bien connu mais, estime le RFC, a quelques défauts. Notamment, obtenir un certificat n'est pas trivial, et certainement bien plus compliqué que de configurer un mot de passe, opération qui est familière (le RFC dit « naturelle », ce qui est exagéré) à beaucoup. La quantité de certificats auto-signés (les plus simples à obtenir) qu'on trouve (surtout sur les MTA) en est la preuve. (Sans compter les autres problèmes comme les certificats expirés <https://www.bortzmeyer.org/tester-expiration-certifs.html>, qui montrent

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5931.txt

bien que l'avitaillement en certificats est un problème non résolu.) Et puis s'authentifier pour obtenir le premier certificat est un problème d'œuf et de poule qu'on pourrait résoudre avec ce nouveau protocole, authentifiant simplement un échange, quitte à obtenir le certificat par la suite, sur le lien sécurisé (méthode décrite dans le RFC 7030). Autre scénario d'usage pour ce nouveau protocole, le cas d'un lecteur qui transmet le PIN à la carte à puce en clair : il serait trop compliqué d'utiliser un certificat dans ce cas, alors qu'une authentification par mot de passe serait plus simple.

Les solutions sans certificat ont des vulnérabilités comme la possibilité d'attaques par dictionnaire. Un attaquant essaie plein de mots de passe possibles (par exemple, le recrutement de Mirai fonctionne ainsi) et tombe forcément juste de temps en temps. Une solution souvent proposée est d'avoir un mot de passe à forte entropie, qui a peu de chances d'être dans les essais effectués par l'attaquant `<https://xkcd.com/936/>`. Par exemple un mot de passe choisi aléatoirement oblige l'attaquant à essayer toutes les possibilités (puisque toutes les combinaisons sont également possibles). Si le mot de passe fait N bits, cela imposera à l'attaquant 2{}(N-1) essais en moyenne.

Mais la faiblesse de ce raisonnement est qu'il ne marche que si l'attaquant peut faire autant d'essais qu'il veut, sans conséquences négatives pour lui (par exemple parce qu'il a mis la main sur un fichier de mots de passe condensés, et qu'il peut à loisir tester hors-ligne s'il en trouve un). D'autres solutions sont pourtant possibles. Ainsi, le PIN d'une carte Visa ne fait que quatre chiffres, ce qui est très peu (5 000 essais suffisent, en moyenne) sauf que la carte se bloque au bout de trois essais échoués. (Notez que sur les très anciennes cartes, il était parfois possible de remettre le compteur d'essais à zéro, annulant cette sécurité.) Ce qui compte, ce ne sont pas les 10 000 possibilités théoriques, ce sont les trois essais. Le RFC généralise cette observation : « la résistance à une attaque par dictionnaire doit être une fonction du nombre d'interactions avec un participant honnête, pas une fonction de la quantité de calculs effectués ». En effet, le nombre d'interactions avec le participant honnête ne croîtra guère dans le futur (l'attaquant est détecté avant) alors que la quantité de calculs réalistiquement possibles croîtra à coup sûr. Comme dans le cas du code PIN, il n'est pas forcément nécessaire d'avoir un mot de passe à forte entropie, il faut juste empêcher l'adversaire d'essayer à volonté, et pour cela le forcer à des attaques actives, interagissant réellement avec sa victime. La section 7 du RFC détaille cette analyse, notant que ce protocole peut se contenter de mots de passe relativement « faibles ». Ainsi, note le RFC, un mot de passe de quatre lettres ASCII peut être suffisant (sans les consignes « votre mot de passe doit comporter au moins douze caractères, dont une lettre minuscule, une lettre majuscule, un chiffre, un emoji et un hiéroglyphe ») puisqu'il offre 459 976 possibilités. Un attaquant qui tenterait la force brute devrait essayer des dizaines de milliers de fois, ce qui sera certainement détecté (par exemple par des logiciels du genre de fail2ban, ou par des IDS).

Désolé, mais je ne vais pas vous exposer le protocole Dragonfly. Il dépasse mes maigres connaissances en cryptographie. La section 3 du RFC expose le minimum qu'il faut savoir en cryptographie pour comprendre la suite (vous aurez besoin de réviser la cryptographie sur courbes elliptiques, par exemple via le RFC 6090 et ça vaut aussi la peine de relire le RFC 8422 sur les extensions TLS spécifiques aux courbes elliptiques). La section 3 rappelle aussi qu'il faut saler les mots de passe avant de les stocker, côté serveur, au cas où le fichier des mots de passe se fasse voler (voir aussi la section 7, qui détaille ce risque et ses conséquences). Le résultat se nomme la **base**. Plus précisement, la base est le résultat de l'application de HMAC-SHA256 à un sel et à la concaténation du nom de l'utilisateur et du mot de passe. Par exemple, si vous voulez le faire avec OpenSSL, que l'utilisateur est "fred" et le mot de passe "barney", avec un sel (tiré aléatoirement) "57ff4d5abdfe2ff3d849fb44848b42f2c41fd995", on fera :

```
%  echo -n fredbarney | openssl dgst -sha256 -hmac "57ff4d5abdfe2ff3d849fb44848b42f2c41fd995" -binary \
       | openssl enc -base64
4m9bv5kWK6t3jUFkF8qk96IuixhG+C6CY7NxsOGrlPw=
```

Comme le mot de passe peut inclure de l'Unicode, il doit être traité selon les règles du RFC 8265. Le serveur va stocker le nom de l'utilisateur, le sel et la base.

C'est la base qui est présentée par le client (après que le serveur lui ait envoyé le sel), pas le mot de passe, et le fichier des bases est donc aussi sensible qu'un fichier de mots de passe en clair (cf. section 7).

La section 4 du RFC décrit le protocole. L'échange de clés utilisé, Dragonfly, est décrit dans le RFC 7664. Utiliser Dragonfly en clair serait sûr du point de vue de l'authentification, mais pas du point de vue de la vie privée puisque le nom d'utilisateur passerait en clair. TLS-PWD, le protocole complet (dont Dragonfly n'est qu'une partie), offre un mécanisme pour protéger contre l'écoute de ce nom, une paire de clés cryptographiques étant générée et utilisée juste pour chiffrer ce bout de la session.

Le protocole TLS-PWD nécessite l'utilisation d'extensions TLS (RFC 5246, notamment la section 7.4.1.2). Elles sont au nombre de trois, notamment `PWD_protect` (protection du nom) et `PWD_clear` (nom d'utilisateur en clair). Elles figurent désormais dans le registre IANA <`https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xml#tls-extensiontype-va`
Une fois le nom reçu et trouvé dans le fichier côté serveur, le client peut s'authentifier avec le mot de passe (c'est plus compliqué que cela : voir la section 4 pour tous les détails).

Dans TLS, la suite complète des algorithmes utilisés dans une session est indiquée dans une variable nommée "*cipher suite*" (RFC 5246, sections 7.4.1.2, 9 et annexe C). Les nouvelles suites permettant l'authentification par mot de passe sont listées dans la section 5 : ce sont `TLS_ECCPWD_WITH_AES_128_GCM_SHA256`, `TLS_ECCPWD_WITH_AES_256_GCM_SHA384`, `TLS_ECCPWD_WITH_AES_128_CCM_SHA256` et `TLS_ECCPWD_WITH_AES_256_CCM_SHA384`. Elles sont indiquées dans le registre IANA <`https://www.iana.org/assignments/tls-parameters/tls-parameters.xml#tls-parameters-4`>.

La section 7 du RFC contient de très nombreux détails sur la sécurité du protocole. Elle note par exemple qu'un attaquant qui essaierait tous les mots de passe d'un même utilisateur serait facilement détecté mais qu'un attaquant pourrait aussi essayer un seul mot de passe avec beaucoup d'utilisateurs, échappant ainsi à la détection. Le RFC recommande donc de compter toutes les tentatives de connexion ensemble, quel que soit l'utilisateur. (Cela traite aussi partiellement le cas d'un IDS extérieur, qui ne verrait pas le nom d'utilisateur, celui-ci étant protégé. L'IDS pourrait néanmoins compter les problèmes de connexion et donner l'alarme.) D'autre part, les mesures contre les attaques par dictionnaire (forcer l'attaquant à interagir avec le serveur, et donc à se signaler) ne sont évidemment pas efficaces si un utilisateur a le même mot de passe sur plusieurs serveurs et que l'un est compromis. La consigne d'utiliser des mots de passe différents par service reste donc valable. (Et, même si le RFC n'en parle pas, il ne faut évidemment pas noter ces mots de passe sur un post-it collé sous le clavier. La relativisation de la règle des mots de passe compliqués permet d'utiliser des mots de passe courts et mémorisables, donc plus d'excuses pour noter sur le post-it.) Sinon, les fanas de cryptographie qui voudraient étudier la sécurité du protocole Dragonfly sont invités à lire l'article de Lancrenon, J. et M. Skrobot, « "*On the Provable Security of the Dragonfly Protocol*" <`https://dl.acm.org/citation.cfm?id=2966298`> ».

Une curiosité : ce RFC est apparemment le premier à avoir une section « Droits Humains » ("*Human Rights Considerations*", section 8). Une telle section, analysant les conséquences du protocole décrit dans le RFC sur les droits humains était suggérée (mais non imposée) par le RFC 8280. Mais, en l'occurrence, celle-ci se réduit à une défense des armes à feu, sans trop de rapport avec le sujet du RFC. On note également une vision très personnelle et très états-unienne des droits humains « le premier des droits est celui de se protéger », ce qui ne figure dans aucun des textes fondateurs des droits humains.