RFC 8922 : A Survey of the Interaction Between Security Protocols and Transport Services

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 22 octobre 2020

Date de publication du RFC : Octobre 2020

https://www.bortzmeyer.org/8922.html

Quels sont les services de sécurité fournis par les protocoles de transport existants ? Ce nouveau RFC fait le point et examine ces services de sécurité pour un certain nombre de ces protocoles, comme TLS, QUIC <https://www.bortzmeyer.org/quic.html>, WireGuard, etc. Cela intéressera tous les gens qui travaillent sur la sécurité de l'Internet.

Ce RFC est issu du groupe de travail IETF TAPS <https://datatracker.ietf.org/wg/taps/>, dont le projet est d'abstraire les services que rend la couche Transport pour permettre davantage d'indépendance entre les applications et les différents protocoles de transport. Vous pouvez en apprendre plus dans les RFC 8095 [1] et RFC 8303.

Parmi les services que fournit la couche Transport aux applications, il y a la sécurité. Ainsi, TCP (si le RFC 5961 a bien été mis en œuvre et si l'attaquant n'est pas sur le chemin) protège assez bien contre les usurpations d'adresses IP <https://www.bortzmeyer.org/usurpation-adresse-ip.html>. Notez que le RFC utilise le terme de « protocole de transport » pour tout ce qui est en dessous de l'application, et utilisé par elle. Ainsi, TLS est vu comme un protocole de transport, du point de vue de l'application, il fonctionne de la même façon (ouvrir une connexion, envoyer des données, lire des réponses, fermer la connexion), avec en prime les services permis par la cryptographie, comme la confidentialité. TLS est donc vu, à juste titre, comme un protocole d'infrastructure, et qui fait donc partie de ceux étudiés dans le cadre du projet TAPS <https://datatracker.ietf.org/wg/taps/>.

La section 1 du RFC explique d'ailleurs quels protocoles ont été intégrés dans l'étude et pourquoi. L'idée était d'étudier des cas assez différents les uns des autres. Des protocoles très répandus comme

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc8095.txt

SSH (RFC 4251), GRE (avec ses extensions de sécurité du RFC 2890) ou L2TP (RFC 5641) ont ainsi été écartés, pas parce qu'ils sont mauvais mais parce qu'ils fournissent à peu près les mêmes services que des protocoles étudiés (par exemple SSH vs. TLS) et ne nécessitaient donc pas d'étude séparée. Plus animée, au sein du groupe de travail TAPS, avait été la discussion sur des protocoles non-IETF comme WireGuard ou MinimaLT <https://www.bortzmeyer.org/minimalt.html>. Ils ne sont pas normalisés (et, souvent, même pas décrits dans une spécification stable), mais ils sont utilisés dans l'Internet, et présentent des particularités suffisamment importantes pour qu'ils soient étudiés ici. En revanche, les protocoles qui ne fournissent que de l'authentification, comme AO (RFC 5925) n'ont pas été pris en compte.

Comme c'est le cas en général dans le projet TAPS, le but est de découvrir et de documenter ce que les protocoles de transport ont en commun, pour faciliter leur choix et leur utilisation par l'application. Par contre, contrairement à d'autres cas d'usage de TAPS, il n'est pas prévu de permettre le choix automatique d'un protocole de sécurité. Pour les autres services, un tel choix automatique se justifie (RFC 8095). Si une application demande juste le service « transport d'octets fiable, j'ai des fichiers à envoyer », lui fournir TCP ou SCTP revient au même, et l'application se moque probablement du protocole choisi. Mais la sécurité est plus compliquée, avec davantage de pièges et de différences subtiles, et il est donc sans doute préférable que l'application choisisse explicitement le protocole.

La section 2 du RFC permet de réviser la terminologie, je vous renvoie à mon article sur le RFC 8303 pour la différence entre fonction ("*Transport Feature*") et service ("*Transport Service*").

Et la section 3 s'attaque aux protocoles eux-mêmes. Elle les classe selon qu'ils protègent n'importe quelle application, une application spécifique, l'application et le transport, ou bien carrément tout le paquet IP. Le RFC note qu'aucune classification n'est parfaite. Notamment, plusieurs protocoles sont séparés en deux parties, un protocole de connexion, permettant par exemple l'échange des clés de session, protocole utilisé essentiellement, voire uniquement, à l'ouverture et à la fermeture de la session, et le protocole de chiffrement qui traite les données transmises. Ces deux protocoles sont plus ou moins intégrés, de la fusion complète (tcpcrypt, RFC 8548) à la séparation complète (IPsec, où ESP, décrit dans le RFC 4303, ne fait que chiffrer, les clés pouvant être fournies par IKE, spécifié dans le RFC 7296, ou par une autre méthode). Par exemple, TLS décrit ces deux protocoles séparement (RFC 8446) mais on les considérait comme liés…jusqu'à ce que QUIC <https://www.bortzmeyer.org/quic.html> décide de n'utiliser qu'un seul des deux.

Bon, assez d'avertissements, passons à la première catégorie, les protocoles qui fournissent une protection générique aux applications. Ils ne protègent donc pas contre des attaques affectant la couche 4 comme les faux paquets RST. Le plus connu de ces protocoles est évidemment TLS (RFC 8446). Mais il y a aussi DTLS (RFC 9147).

Ensuite, il y a les protocoles spécifiques à une application. C'est par exemple le cas de SRTP (RFC 3711), extension sécurisée de RTP qui s'appuie sur DTLS.

Puis il y a la catégorie des protocoles qui protègent le transport ce qui, par exemple, protège contre les faux RST ("*ReSeT*"), et empêche un observateur de voir certains aspects de la connexion. C'est le cas notamment de QUIC <https://www.bortzmeyer.org/quic.html>, normalisé depuis. QUIC utilise TLS pour obtenir les clés, qui sont ensuite utilisées par un protocole sans lien avec TLS, et qui chiffre y compris la couche Transport. Notons que le prédécesseur de QUIC, développé par Google sous le même nom (le RFC nomme cet ancien protocole gQUIC pour le distinguer du QUIC normalisé), n'utilisait pas TLS.

Toujours dans la catégorie des protocoles qui protègent les couches Transport et Application, tcpcrypt (RFC 8548). Il fait du chiffrement « opportuniste » (au sens de « sans authentification ») et est

donc vulnérable aux attaques actives. Mais il a l'avantage de chiffrer sans participation de l'application, contrairement à TLS. Bon, en pratique, il n'a connu quasiment aucun déploiement.

Il y a aussi MinimaLT <https://www.bortzmeyer.org/minimalt.html>, qui intègre l'équivalent de TCP et l'équivalent de TLS (comme le fait QUIC).

Le RFC mentionne également CurveCP <http://curvecp.org>, qui fusionne également transport et chiffrement. Comme QUIC ou MinimaLT, il ne permet pas de connexions non-sécurisées.

Et enfin il y a les protocoles qui protègent tout, même IP. Le plus connu est évidemment IPsec (RFC 4303 et aussi RFC 7296, pour l'échange de clés, qui est un protocole complètement séparé). Mais il y a aussi WireGuard qui est nettement moins riche (pas d'agilité cryptographique, par exemple, ni même de négociation des paramètres cryptographiques ; ce n'est pas un problème pour des tunnels statiques mais c'est ennuyeux pour un usage plus général sur l'Internet). Et il y a OpenVPN, qui est sans doute le plus répandu chez les utilisateurs ordinaires, pour sa simplicité de mise en œuvre. On trouve par exemple OpenVPN dans tous des systèmes comme OpenWrt et donc dans tous les routeurs qui l'utilisent.

Une fois cette liste de protocoles « intéressants » établie, notre RFC va se mettre à les classer, selon divers critères. Le premier (section 4 du RFC) : est-ce que le protocole dépend d'un transport sous-jacent, qui fournit des propriétés de fiabilité et d'ordre des données (si oui, ce transport sera en général TCP) ? C'est le cas de TLS, bien sûr, mais aussi, d'une certaine façon, de tcpcrypt (qui dépend de TCP et surtout de son option ENO, normalisée dans le RFC 8547). Tous les autres protocoles peuvent fonctionner directement sur IP mais, en général, ils s'appuient plutôt sur UDP, pour réussir à passer les différentes "*middleboxes*" qui bloquent les protocoles de transport « alternatifs ».

Et l'interface avec les applications ? Comment ces différents protocoles se présentent-ils aux applications qui les utilisent ? Il y a ici beaucoup à dire (le RFC fournit un tableau synthétique de quelles possibilités et quels choix chaque protocole fournit aux applications). L'analyse est découpée en plusieurs parties (les services liés à l'établissement de la connexion, ceux accessibles pendant la session et ceux liés à la terminaison de la connexion), chaque partie listant plusieurs services. Pour chaque service, le RFC dresse la liste des différents protocoles qui le fournissent. Ainsi, la partie sur l'établissement de connexion indique entre autres le service « Extensions au protocole de base accessibles à l'application ». Ce service est fourni par TLS (via ALPN, RFC 7301) ou QUIC mais pas par IPsec ou CurveCP. De même le service « Délégation de l'authentification » peut être fourni par IPsec (par exemple via EAP, RFC 3748) ou tcpcrypt mais pas par les autres.

La section 7 rappelle que ce RFC ne fait qu'analyser les protocoles existants, il ne propose pas de changement. D'autre part, cette section note que certaines attaques ont été laissées de côté (comme celle par canal secondaire ou comme l'analyse de trafic).

La section 8 sur la vie privée rappelle ces faiblesses ; même si l'un des buts principaux du chiffrement est d'assurer la confidentialité, la plupart des protocoles présentés laissent encore fuiter une certaine quantité d'information, par exemple les certificats en clair de TLS (avant la version 1.3).