RFC 3552 : Guidelines for Writing RFC Text on Security Considerations

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 27 mai 2009

Date de publication du RFC : Juillet 2003

https://www.bortzmeyer.org/3552.html

La sécurité est désormais devenue une préoccupation constante de tous les administrateurs réseau et système. Elle concerne également les auteurs de logiciels. Mais les concepteurs de protocoles ne sont pas épargnés. L'IETF demande désormais à tout protocole conçu pour un déploiement sur l'Internet d'être impeccable, sur le plan de la sécurité, d'avoir prévu les attaques possibles, et les parades à utiliser. Pour éviter tout oubli, il est obligatoire, dans tout RFC, d'avoir une section "*Security considerations*", qui contient l'analyse des questions de sécurité du protocole normalisé par ce RFC. Que mettre dans cette section ? C'est le but de ce RFC 3552 [1] que de l'expliquer.

Ce RFC est donc une lecture indispensable pour l'auteur de RFC. Il porte toute l'autorité de l'IAB. Mais il est aussi un excellent tutoriel sur la sécurité sur l'Internet, notamment par son exposé des différents types d'attaques, et il est donc une lecture recommandée pour toute personne qui débute sur ce sujet. Elle y apprendra le vocabulaire et les concepts de base de la sécurité, qui font défaut à tant d'administrateurs aujourd'hui.

À quoi sert la section Sécurité des RFC, obligatoire depuis le RFC 2223 (section 9) ? Comme le rappelle la section 1 de notre RFC 3552, cette section Sécurité sert à s'assurer que les auteurs du protocole ont procédé à une analyse sérieuse de la sécurité de leur œuvre, et à informer les lecteurs des résultats de ladite analyse.

Bon, mais avant de prendre des mesures de sécurité, il faut mieux définir ce qu'est la sécurité. En avant pour la section 2, qui introduit la partie « tutoriel » du RFC. La sécurité est en fait un ensemble de propriétés diverses. On peut identifier deux catégories, la sécurité des communications et celles des systèmes qui communiquent.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc3552.txt

La sécurité des communications est détaillée en section 2.1. Elle se subdivise entre trois sous-catégories, la **confidentialité**, l'**intégrité des données** et l'**authentification** du partenaire. La confidentialité est le fait que vos secrets restent secrets : personne ne doit pouvoir écouter vos communications. L'intégrité des données est le fait qu'on reçoit bien ce qui a été envoyé, aucun méchant n'a modifié ces données (un objectif très difficile à atteindre dans le monde numérique : contrairement au papier, les changements ne laissent pas de trace). Et l'authentification est le fait de s'assurer qu'on parle bien avec celui qu'on croit.

Ces trois propriétés sont liées : par exemple, sans authentification, la confidentialité est difficile à assurer (si Alice sécurise ses communications contre l'écoute, mais n'authentifie pas, qui sait si elle n'écrit pas à Ève en pensant écrire à Bob ? Auquel cas, ses précautions ne serviraient à rien.)

Une autre propriété qui ne rentre pas dans cette classification est la non-répudiation (section 2.2) qui est le fait de pouvoir prouver que le message a bien été émis.

La sécurité des systèmes est en section 2.3. Elle comprend entre autres :
— La protection contre l'usage non autorisé du système,
— La protection contre les usages inappropriés (les petites lettres dans les contrats qui limitent ce que vous avez le droit de faire),
— La protection contre les dénis de service, les attaques qui visent à empêcher tout usage de la machine.

Avec la section 3, on étudie cette fois les méchants, les agresseurs. Elle est consacrée à la modélisation des menaces. L'idée est qu'on ne peut pas se protéger contre tout (les meilleures protections contre le déni de service, par exemple, ne serviront pas si l'attaquant coupe physiquement le câble <http://www.wired.com/threatlevel/2009/04/cable-sabotage/>). La modélisation des menaces la plus courante sur l'Internet consiste à considérer que l'attaquant :
— N'a pas pris le contrôle d'une des deux machines qui communique (autrement, la défense devient quasi-impossible),
— Mais, en revanche, a un contrôle total du réseau qui les relie. Il peut lire n'importe quel paquet, en modifier et en fabriquer, y compris avec une adresse IP source qui n'est pas la sienne.

Il existe aussi des modèles de menaces plus limités (section 3.1) par exemple où l'attaquant est strictement passif : s'il peut lire n'importe quel paquet, il n'a pas la possibilité d'en créer ou d'en modifier. De telles attaques représentent une proportion non négligeable, dans le monde réel. (En sécurité des réseaux, Ève est un attaquant passif, Mallory un attaquant actif.)

La section 3.2 détaille donc ce que peut faire un attaquant **passif**. Les attaques passives nécessitent d'être quelque part sur le chemin entre les deux partenaires (section 3.5 pour cette notion d'« être sur le chemin »), par exemple parce qu'on est sur le même réseau Wifi (réseau partagé, voir aussi section 3.6) qu'une des deux machines qui communiquent, ou bien parce qu'on contrôle un routeur sur le chemin (beaucoup de routeurs sont mal protégés <http://www.arbornetworks.com/en/docman/worldwide-infrastructure-security-report-volume-iv-2008-/download.html>). Dans ces conditions, l'attaquant passif peut :
— Écouter les communications,
— Capturer des mots de passe (par exemple avec l'excellent outil dsniff),
— Faire, avec les données capturées, des attaques **hors-ligne**, par exemple en cryptanalysant.

Passons aux attaques actives. La section 3.3 leur est consacrée. Un attaquant actif peut fabriquer des paquets IP quelconques (en théorie, les FAI ne devraient pas laisser passer les paquets ayant une adresse IP source usurpée, mais le RFC 2827 est bien peu déployé). Il ne peut pas toujours recevoir la réponse (qui ira à celui dont il a usurpé l'adresse) mais certaines attaques, dites **aveugles**, sont malgré tout possibles dans ce cas.

Parmi les autres attaques possibles, il y a le rejeu (section 3.3.1) où l'attaquant réemet un paquet qu'il a déjà vu (sans même le comprendre, par exemple si le message déclenche un retrait bancaire, et que le protocole n'a pas de protection contre le rejeu, on voit ce qu'il peut advenir), la modification de message

(section 3.3.4) ou l'homme du milieu où Janus, se plaçant entre Alice et Bob, se fait passer pour Bob auprès d'Alice et pour Alice auprès de Bob (et relaie les messages, pour qu'Alice et Bob ne s'aperçoivent de rien).

Maintenant, arrivé à ce stade, on connait les agresseurs et leurs capacités. Il existe évidemment de nombreuses défenses possibles mais toutes ne sont pas bonnes, loin de là. La section 4 doit donc lister des questions fréquentes, communes à beaucoup de protocoles. Par exemple, pour l'authentification des utilisateurs (section 4.1), notre RFC rappelle les forces et les faiblesses des différents mécanismes, comme le traditionnel mot de passe (désormais interdit dans les protocoles IETF, sauf si le canal est protégé par la cryptographie, le risque de "*sniffing*" étant trop élevé autrement) ou les certificats (RFC 5280), qui peuvent être également utilisés pour authentifier une machine.

En fait, concevoir un protocole sûr est une tâche tellement complexe, avec tellement de possibilités d'erreur (et, en sécurité, les erreurs ne se voient pas, avant qu'elles ne soient exploitées), que la meilleure solution est souvent... de ne pas le faire, mais de compter sur un cadre générique (section 4.2) comme SASL (RFC 2222) pour l'authentification. En disant simplement « Pour l'authentification, on utilise SASL », l'auteur de protocole prudent peut ainsi s'appuyer sur un cadre solide et prouvé. Ceci dit, rien n'est jamais simple en sécurité et le RFC 3552 rappelle que ces cadres génériques ont souvent une faiblesse, ils peuvent être victimes d'une **attaque par repli**, où un attaquant actif va essayer de faire en sorte qu'Alice et Bob n'utilisent pas la sécurité maximale, si d'autres sont disponibles. Bien sûr, tout protocole de sécurité où les paramètres sont négociables court ce risque mais il est plus élevé pour les cadres génériques car ils encouragent le développeur du protocole à ne pas trop réfléchir aux détails.

La section 4.4 rappelle la différence entre authentifier et autoriser <https://www.bortzmeyer.org/authentifier-et-autoriser.html> (le premier permet de vérifier une identité, le second de déterminer si une entité donnée a le droit de faire telle action).

Comment fournir des communications sécurisées entre deux machines ? Un protocole donné a deux solutions, fournir lui-même cette sécurité ou bien s'appuyer sur un protocole de sécurité généraliste. Ainsi, le DNS se protège contre les modifications en transit par la signature des enregistrements (DNSSEC, RFC 4033), la première solution, celle d'une sécurité fournie par le protocole. Par contre, HTTP délègue en général sa sécurité à un protocole générique, TLS (RFC 5246 et RFC 2818 pour l'adaptation spécifique à HTTP), ce qui est la deuxième solution. La section 4.5 de notre RFC discute ces deux solutions et donne des exemples.

Par exemple, le mécanisme le plus générique est IPsec (RFC 4301). Cette solution protège toute communication IP. Mais IPsec est peu déployé, et en général sous forme de VPN. Un inconvénient de la méthode VPN est qu'elle est invisible aux applications, ce qui rend difficile pour celles-ci de savoir si leurs communications sont protégées (et donc si elles peuvent employer un mot de passe classique). Le RFC demande donc que les auteurs de protocole ne tiennent pas pour acquise la disponibilité d'IPsec (une violation de cette règle se trouve dans le RFC 5340, qui a supprimé les possibilités d'authentification qui existaient dans les précédentes versions d'OSPF, pour imposer IPsec). Pourtant, le RFC rappelle qu'IPsec est obligatoire pour IPv6 (ce qui est de la pure réthorique, la plupart des implémentations d'IPv6 n'ont pas IPsec et, même quand il est implémenté, son activation reste facultative... et compliquée).

L'autre grand mécanisme générique de protection des communications est TLS (RFC 5246) qui travaille, lui, juste en dessous de la couche 7. Il ne protège donc pas contre, par exemple, des paquets TCP RST ("*ReSeT*") faux. Notre RFC note qu'il existe deux façons d'utiliser TLS, avec un port spécial dédié (ce que fait HTTPS, avec son port 443) ou bien avec un démarrage de TLS une fois la connexion établie. À part HTTP, où le RFC 2817 n'a eu aucun succès, cette seconde méthode est la plus employée parmi les protocoles TCP/IP, sous le nom de `STARTTLS`. Son avantage est qu'elle permet de

n'utiliser qu'un seul port et de négocier certains paramètres avant de lancer TLS et donc de choisir le certificat en fonction de critères comme le nom demandé (ce qui est utile en HTTP, autrement, avoir plusieurs certificats sur une même machine est complexe <https://www.bortzmeyer.org/plusieurs-noms-dans-certificat.html>, cf. section 4.5.2.1). Son inconvénient principal est qu'elle peut ouvrir la voie à des attaques par repli, où Mallory essaie de convaincre Alice et Bob que TLS n'est pas possible, les amenant à ne pas sécuriser leur communication.

Les attaques par déni de service forment une classe entière d'attaques, très difficiles à combattre. Très fréquentes sur l'Internet aujourd'hui (un exemple est l'attaque contre les serveurs racine du DNS <https://www.bortzmeyer.org/attaque-serveurs-racine.html> en février 2007), elles n'ont pas de solution générale, contrairement à la plupart des autres attaques où la cryptographie résout tous les problèmes techniques. La section 4.6 étudie cette menace et les approches possibles. Certaines attaques par déni de service sont brutales et sans subtilité, d'autres plus avancées comme les attaques "*SYN flood*" <http://www.cert.org/advisories/CA-1996-21.html> contre TCP. Il est parfois possible de modifier les protocoles pour les rendre moins vulnérables aux attaques subtiles. Par contre, pour les attaques par force brute, les seules protections restent le surdimensionnement des infrastructures (si c'est économiquement raisonnable) et la réaction "*a posteriori*".

Parmi les variétés de DoS, le RFC signale l'attaque aveugle (comme le "*TCP SYN flood*" cité plus haut), où l'attaquant n'a même pas besoin de pouvoir recevoir les réponses. Mais il y a aussi la DoS distribuée où des dizaines ou des centaines de milliers de machine MS-Windows, devenues des zombies, coopèrent pour l'attaque.

Quant aux protections, elles reposent sur les principes suivants :

— Faire en sorte que l'attaquant aie plus de travail que vous (section 4.6.3.1) par exemple en demandant au client de résoudre des puzzles cryptographiques comme le fait HIP (RFC 5201). Cela ne protège pas contre les attaques distribuées, où l'ennemi a une énorme puissance de calcul à sa disposition.
— Obliger l'attaquant à montrer qu'il peut recevoir les réponses par exemple en générant aléatoirement un **petit gâteau**, un ensemble de bits que le partenaire va devoir transmettre "*verbatim*", prouvant ainsi qu'il reçoit les réponses. Cela empêche les attaques aveugles et valide ainsi l'adresse IP source de l'attaquant. (Photuris, RFC 2522 utilise cette technique.)

Il y a une autre façon de voir les techniques de sécurité, c'est de séparer celles qui protègent le **message** et celles qui protègent le **canal**. Dans le premier cas, le message est protégé de bout en bout, quels que soient les intermédiaires. Dans le second, on ne protège qu'un échange entre deux machines, mais des intermédiaires ont pu lire et/ou modifier le message avant ou après le canal sécurisé. Par exemple, pour le DNS, DNSSEC fournit une protection de bout en bout, il protège le message. Même si un des serveurs de noms secondaires du domaine ou un serveur de noms récursif est sous le contrôle d'un attaquant, DNSSEC permettra de détecter une modification. Au contraire, DNSCurve ne protège que le canal. Si un serveur de noms récursif triche, DNSCurve ne protège pas (le point n'est pas mentionné sur le site Web de DNSCurve, ce qui n'est pas étonnant, l'honnêteté intellectuelle n'ayant jamais été le fort de djb). Autre exemple, avec le courrier électronique, PGP (RFC 4880) sécurise le message (et il est donc protégé contre la lecture non autorisée ou la modification, quels que soient le comportement des serveurs intermédiaires) alors que SMTP + TLS (RFC 3207) ne protège qu'un canal (alors que le courrier est relayé, stocké, etc). Le RFC note que la distinction entre protection du message et protection du canal dépend de la couche qu'on regarde. Au niveau IP, chaque paquet est un message et donc IPsec peut dire que, au niveau 3, il sécurise le message alors que, au niveau 7, les applications considèrent qu'IPsec ne sécurise que le canal. Et il y a des cas encore plus compliqués comme une page Web dont les différents composants ont été récupérés par des canaux différents.

Enfin, après toutes ses considérations sur la sécurité des réseaux en général et de l'Internet en particulier, on arrive avec la section 5 aux prescriptions sur la section "*Security considerations*" des futurs RFC. Que faut-il y mettre ? Les auteurs **doivent** indiquer :

— quelles attaques sont exclues par l'analyse de sécurité (et pourquoi !)
— quelles attaques sont inclues dans l'analyse et, pour chacune, si le protocole est protégé contre cette attaque ou pas.

Le RFC doit également inclure un modèle des menaces, s'appliquant à l'Internet entier (pas uniquement à un réseau local déconnecté du reste du monde). Un exemple que donne le RFC concerne l'authentification traditionnelle Unix via le mot de passe et les fichiers `/etc/passwd` et `/etc/shadow` et un texte possible serait quelque chose du genre « La sécurité du système repose sur la difficulté à deviner le mot de passe et à écouter le réseau lorsque le mot de passe circule. La divination du mot de passe est possible hors-ligne avec des programmes comme Crack (surtout si la longueur du mot est limitée à huit caractères) et l'écoute est possible dès que le mot de passe est utilisé en clair, par exemple par telnet. »

Enfin, la section 6 fournit des exemples de « bonnes » sections Sécurité, tirées de RFC réels comme le RFC 2821 (avec des ajouts mis par le RFC 3552 pour traiter des problèmes que le RFC 2821 avait négligé) ou comme le RFC 2338, là encore avec quelques ajouts (le RFC 2338 oubliait de dire explicitement que la confidentialité n'était pas un but).