RFC 8827 : WebRTC Security Architecture

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 19 janvier 2021

Date de publication du RFC : Janvier 2021

https://www.bortzmeyer.org/8827.html

Le système WebRTC permet des communications audio et vidéo entre deux navigateurs Web. Comme tout service Internet, il pose des problèmes de sécurité (RFC 8826 [1]), et ce RFC étudie l'architecture générale de WebRTC, du point de vue de la sécurité, et comment résoudre les problèmes.

D'abord, il faut réviser les généralités sur WebRTC (RFC 8825). WebRTC a pour but de permettre à deux utilisateurs ou utilisatrices de navigateurs Web de s'appeler et de bavarder en audio et vidéo. Pas besoin d'installer un logiciel de communication spécifique (comme le Skype de Microsoft, ou comme un "*softphone*" tel que Twinkle), il suffit d'un navigateur ordinaire. Pour permettre cela, WebRTC repose sur une API JavaScript normalisée par le W3C et sur un ensemble de protocoles normalisés par l'IETF, et résumés dans le RFC 8825. Si on prend l'exemple du service WebRTC FramaTalk, Alice et Bob visitent tous les deux , récupèrent du code JavaScript qui va utiliser l'API WebRTC pour accéder au micro et à la caméra, pendant que, derrière, FramaTalk va établir une communication entre eux, permettant à leurs navigateurs d'échanger du son et des images. Cet établissement de connexion (qui peut impliquer plusieurs serveurs, s'il y a fédération) n'est **pas** spécifié par WebRTC. (Il n'y a pas automatiquement d'interopérabilité entre services WebRTC.) Cela peut utiliser SIP (RFC 3261), par exemple.

Cela soulève plein de problèmes de sécurité amusants, étudiés dans le RFC 8826, et que notre RFC 8827 tente de résoudre.

Toute solution de sécurité dépend d'un **modèle de confiance** (section 3 de notre RFC). Si on ne fait confiance à rien ni personne, on ne peut résoudre aucun problème de sécurité. Pour WebRTC, la racine de toute confiance est le navigateur. C'est lui qui devra faire respecter un certain nombre de règles. (En conséquence de quoi, si le navigateur n'est pas de confiance, par exemple parce qu'il s'agit de logiciel non libre, ou bien parce que vous êtes sur une machine sur laquelle vous n'avez aucun contrôle, par exemple un cybercafé, tout est fichu.) Mais le navigateur ne suffit pas, il faut également faire confiance à des entités extérieures, comme le site Web où vous allez récupérer le code JavaScript. Certaines de ces entités peuvent être authentifiées par le navigateur, ce qui donne certaines garanties. Et d'autres, hélas, ne le peuvent pas. Dans la première catégorie, on trouve :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc8826.txt

— Les sites Web où le navigateur vérifie un certificat, dans une connexion HTTPS. Vous vous connectez à (notez le `https`), vous êtes sûr que c'est bien Framasoft derrière.
— Les pairs WebRTC authentifiés par une technique comme DTLS-SRTP.

Le reste des entités avec qui on communique est considéré comme potentiellement malveillant, et il faut donc se méfier (mais, évidemment, dans la vie, on ne communique pas qu'avec des gens qu'on connait bien et à qui on fait confiance : ce serait trop limité).

Pour revenir aux entités authentifiées, notez qu'on n'a parlé que d'authentification, pas de confiance. La cryptographie permet de vérifier qu'on parle bien au diable, pas que le diable est honnête !

La section 4 résume le fonctionnement général de WebRTC, afin de pouvoir analyser sa sécurité, et définir une architecture de sécurité. Notez que WebRTC permet pas mal de variantes et que le schéma présenté ici est un cas « typique », pas forcément identique partout dans tous ses détails. Donc, ici, Alice et Bob veulent se parler. Tous les deux utilisent un navigateur. Chacun va visiter le site Web du service qu'ils utilisent, mettons `https://meet.example/` (en HTTPS, donc, avec authentification via un certificat). Ils récupèrent un code JavaScript qui, appelant l'API WebRTC, va déclencher l'envoi d'audio et de vidéo, transportés sur SRTP, lui-même sur DTLS (les données en dehors du « multimédia » passent sur du SCTP sur DTLS). Avant d'autoriser l'accès au micro et à la caméra, chaque navigateur demandera l'autorisation à son utilisat·eur·rice. Quant au fait que la machine en face accepte de recevoir (en direct, en pair-à-pair) ce déluge de données multimédia, la vérification sera faite par ICE (cf. RFC 8826, section 4.2). ICE n'intervient qu'à la connexion initiale, des messages périodiques seront nécessaires pendant la communication pour s'assurer qu'il y a toujours consentement à recevoir. Pendant l'échange de données audio et vidéo, le site Web `https://meet.example/` va assurer la signalisation et, par exemple, si Alice part, indiquer à Bob qu'il n'y a plus personne en face (notez que cet échange entre site Web et utilisateur ne se fait pas avec un protocole normalisé). Notre RFC ajoute une possibilité : qu'Alice et Bob utilisent un fournisseur d'identité, permettant à chacun d'identifier et d'authentifier l'autre (par exemple via OpenID Connect).

Ce cas où les deux participant·e·s partagent le même serveur est le cas le plus courant aujourd'hui (c'est ce que vous verrez si vous faites une visioconférence via). Le RFC le complique ensuite légèrement en introduisant de la fédération : Alice et Bob se connectent à des sites Web différents, mais qui ont un mécanisme de fédération commun. Les deux sites vont alors devoir s'échanger de l'information de signalisation, utilisant des protocoles existants comme SIP ou XMPP (qui n'étaient pas utilisés dans le cas initial).

Armés de ces informations, nous pouvons lire la section 6, qui plonge dans les détails techniques. D'abord, la securité de la connexion entre navigateur et site Web. Elle dépend de HTTPS (RFC 2818), de la vérification du certificat, bien sûr (RFC 6125 et RFC 5280) et de la vérification de l'origine (RFC 6454). La sécurité entre les pairs qui communiquent impose également l'usage de la cryptographie et il faut donc utiliser SRTP (RFC 3711) sur DTLS (RFC 9147 et RFC 5763). Pas question de faire du RTP en clair directement sur UDP ! Le niveau de sécurité (par exemple les paramètres TLS choisis) doivent être accessibles à l'utilisateur.

Ensuite, la securité de l'accès au micro et à la caméra. Le site Web qu'on visite peut être malveillant, et le navigateur ne permet évidemment pas à du code JavaScript de ce site d'ouvrir silencieusement la caméra. Il demande donc la permission à l'utilisateur, qui ne la donne que s'il était vraiment en train de passer un appel. De même, le navigateur doit afficher si micro et caméra sont utilisés, et ne doit pas permettre à du code JavaScript de supprimer cet affichage.

Dans WebRTC, l'échange des données audio et vidéo se fait en pair-à-pair, directement entre Alice et Bob. Il y a donc le risque qu'un méchant envoie plein de données vidéo à quelqu'un qui n'a rien demandé (la section 9.3 détaille le risque de déni de service). Il faut donc vérifier le consentement à

recevoir. Comme indiqué plus haut, cela se fait avec ICE (RFC 8445). Comme le consentement initial n'est pas éternel, et comme on ne peut pas se fier aux accusés de réception, que DTLS n'envoie pas, ni aux réponses (il n'y en a pas forcément, par exemple si on regarde juste une vidéo), il faut envoyer de temps en temps des nouveaux messages ICE (cf. RFC 7675).

Parmi les questions de sécurité liées à WebRTC, il en est une qui a fait beaucoup de bruit, celle de la vie privée, notamment le problème de l'adresse IP qui est transmise au pair avec qui on communique. (Notez que le serveur du site Web où on se connecter initialement verra toujours, lui, cette adresse IP, sauf si on utilise une technique comme Tor.) Notre RFC impose donc, en cas d'appel entrant, de ne pas commencer la négociation ICE avant que l'utilisatrice n'ait décidé de répondre à l'appel, et que WebRTC permette de décider de n'utiliser que TURN, qui ne communique pas l'adresse IP au pair puisqu'on ne se parle plus directement, mais via le relais TURN. Évidemment, cela se fait au détriment des performances (les détails sont dans la section 9.2).

On a vu plus haut que les deux pairs qui communiquent peuvent désirer s'authentifier (sans se fier au site Web qu'ils utilisent pour la signalisation), par exemple via des fournisseurs d'identité externes (comme FranceConnect ?) C'est notamment indispensable pour une fédération puisque, dans ce cas, on n'a même plus de site Web commun. La section 7 de notre RFC détaille ce cas. WebRTC ne normalise pas une technique unique (comme OAuth) pour cela (WebRTC est un cadre, pas vraiment un protocole, encore moins un système complet), mais fixe quelques principes. Notamment, l'utilisateur s'authentifie auprès du fournisseur d'identité, pas auprès du site Web de signalisation. Le navigateur Web doit donc faire attention à tout faire lui-même et à ne pas laisser ce site Web interférer avec ce processus.

Enfin, il reste des problèmes de sécurité de WebRTC qui ne sont pas couverts par le RFC 8826 ni par les précédentes sections de notre RFC 8827. Ainsi, il est important que le code JavaScript téléchargé depuis le site Web du fournisseur de service WebRTC ne puisse pas directement fabriquer des messages chiffrés et signés pour DTLS, car il pourrait alors fabriquer de vraies/fausses signatures. Tout doit passer par le navigateur qui est le seul à pouvoir utiliser les clés de DTLS pour chiffrer et surtout signer.

Une bonne analyse de la sécurité de WebRTC se trouve en « "*A Study of WebRTC Security*" `<http://webrtc-security.github.io/#4.5>` ».