RFC 7687 : Report from the Strengthening the Internet (STRINT) workshop

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 15 décembre 2015

Date de publication du RFC : Décembre 2015

https://www.bortzmeyer.org/7687.html

Depuis les révélations d'Edward Snowden, les initiatives se multiplient pour « durcir l'Internet », c'est-à-dire le rendre plus résistant à la surveillance généralisée, telle que pratiquée par plusieurs pays (par exemple la France, où les politiques profitent cyniquement des attentats islamistes pour faire passer de plus en plus de lois liberticides). Début 2014, à Londres, s'est ainsi tenu l'atelier STRINT <https://www.w3.org/2014/strint/Overview.html>, qui a rassemblé une centaine de participants pour réfléchir à la meilleure façon de renforcer l'Internet contre cet espionnage. Ce RFC est le compte-rendu (très tardif) de cet atelier. Il résume les principales interventions, et les suggestions faites à cette occasion.

L'engagement de l'IETF à lutter contre la surveilance généralisée date de sa réunion de Vancouver <https://www.bortzmeyer.org/ietf-securite-espionnage-bis.html>, et a été formalisé dans le RFC 7258 [1]. C'est à la suite de cette réunion canadienne qu'a été tenu l'atelier STRINT <https://www.w3.org/2014/strint/Overview.html>, coorganisé par le projet STREWS <http://www.strews.eu/>, l'IAB et le W3C. STRINT signifie « "*Strengthening the Internet Against Pervasive Monitoring*" » et l'atelier s'est tenu juste avant la réunion IETF 89 <http://www.ietf.org/blog/2014/03/ietf-89-summary/> à Londres.

L'atelier a été un succès. Prévu pour un petit nombre de participants (cent maximum : le but était de travailler, pas de faire des discours devant une immense salle), il a été rempli en peu de temps. (Outre les 100 participants physiques, jusqu'à 165 personnes ont suivi l'atelier à distance.) 67 articles avaient été soumis.

La section 2 du RFC résume les conclusions de l'atelier :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7258.txt

— La cryptographie **bien faite** est une protection efficace contre la surveillance massive et son usage plus intense est souhaitable.
— L'analyse de trafic est une menace réelle, et la cryptographie est ici moins efficace. Mais le problème reste insuffisamment connu et mérite d'avantage d'études.
— La surveillance massive est quelque chose de relativement nouveau et mérite des analyses du modèle de menace plus détaillées (le travail était en cours à l'époque, et cela a mené, entre autres, au RFC 7624).
— Cette surveillance massive mérite qu'on s'attaque à une mise à jour du RFC 3552.
— Le terme de « "*opportunistic security*" » a été brandi plusieurs fois pendant l'atelier. Il faut savoir que ce terme n'a pas de définition rigoureuse et est utilisé dans le monde de la sécurité pour désigner des tas de choses différentes. Depuis, l'IETF a publié sa définition, dans le RFC 7435.
— Un effort d'explication auprès des décideurs reste nécessaire, pour expliquer les problèmes techniques que pose la surveillance massive. (Belle naïveté de techniciens honnêtes qui croient que les politiques prennent des mauvaises décisions parce qu'ils ne sont pas bien informés <https://www.bortzmeyer.org/pas-sage-en-seine-politiques.html>.)
— La question des interfaces utilisateur est très importante et très difficile. Il existe désormais un net consensus chez les gens de la sécurité que l'utilisateur est un maillon crucial (et souvent fragile) de la chaîne de sécurité. Il reste à tirer des conséquences pratiques de ce consensus de principe. Par exemple, faut-il arrêter de présenter à l'utilisateur des choix dangereux, comme « le certificat n'a pas le bon nom, voulez-vous continuer ? » Si on répond « il ne faut pas proposer de tels choix », une coordination entre les auteurs de logiciels est nécessaire autrement ceux qui respectent le plus la sécurité risquent d'être considérés comme moins bon que ceux de leurs concurrents (comme on l'a déjà vu avec DNSSEC <https://www.bortzmeyer.org/dnssec-qui-est-coupable.html>). Globalement, ce problème des interfaces utilisateur reste un champ de recherche ouvert.
— Développer des exemples de configurations sûres à copier/coller pour mettre dans son logiciel aiderait certainement (regardez par exemple la documentation de l'ANSSI sur HTTPS <https://www.ssi.gouv.fr/guide/recommandations-pour-la-securisation-des-sites-web/>, section 2.1). Ce n'est pas un travail de normalisation et cela ne concerne donc pas directement l'IETF ou le W3C mais c'est certainement important.
— Plus directement de la responsabilité de ces SDO : faire un effort pour que les protocoles soient sûrs par défaut (un contre-exemple est SMTP, qui est en clair par défaut.)
— Apparemment plus anecdotique, mais néanmoins crucial vu leur vaste déploiement, résoudre le problème des portails captifs aiderait : aujourd'hui, ces portails utilisent les mêmes techniques que des attaquants et sont donc indistinguables d'une attaque.

La section 3 de notre RFC revient sur les buts de l'atelier. Une fois qu'on a décidé que la surveillance généralisée était une attaque, et qu'il fallait la combattre (RFC 7258), que faire de plus ?

— Se mettre d'accord sur des concepts comme celui de « sécurité opportuniste »,
— Mieux comprendre les compromis à faire (la sécurité n'est pas gratuite),
— Identifier les « maillons faibles » de la sécurité du Web,
— Et surtout, identifier les tâches concrètes qui relèvent des organismes de normalisation comme l'IETF ou le W3C et celles qui, sans relever directement de ces organismes, les aideraient dans leur tâche.

Pour atteindre ces buts, l'atelier était structuré pour maximiser les discussions (section 4 du RFC). Les papiers acceptés n'étaient pas présentés (les participants étaient supposés les avoir lus avant).

Quels furent les sujets couverts dans les sessions de l'atelier (section 5 du RFC) ? La session d'ouverture a discuté de questions plutôt « méta » comme la quantité minimale de métadonnées qui est vraiment nécessaire aux protocoles IETF pour faire leur travail, comme la recherche de « fruits à portée de main » (des solutions qui pourraient être déployées très rapidement), ou comme le niveau de sécurité considéré comme « suffisant » (il ne faut pas espérer atteindre une sécurité parfaite, quand l'attaquant a les moyens de la NSA).

La première session portait (section 5.2) sur les menaces. Pas de sécurité sérieuse sans une étude détaillée des menaces. Ce n'est pas la même chose de se protéger contre la DGSE ou le FSB et de se

protéger contre le lycéen boutonneux dans son garage. La surveillance généralisée met en cause un modèle traditionnel de menaces, fondé sur une étude coût/bénéfice de la défense (ou de l'attaque d'ailleurs). Dans ce modèle traditionnel, on regardait ce que valait l'objectif à défendre et on calculait les dépenses de sécurité réalistes en fonction de cela. (L'attaquant faisait un calcul similaire, et pouvait migrer vers un autre objectif, si le premier envisagé était trop coûteux.) Mais, avec la surveillance généralisée, il n'y a plus de décisions d'attaquer tel ou tel objectif : on attaque tout le monde. Et l'attaquant étant en général un État, il ne fait pas de calcul de ROI (certaines bureaucraties étatiques sont même contentes quand c'est cher, cela permet de se « construire un empire » en réclamant des budgets toujours plus élevés).

Est-ce que la surveillance généralisée est plus facile s'il y a un petit nombre de silos très protégés (les GAFA, Google, Facebook, etc) ou s'il y a une multitude de petits sites, chacun avec son CozyCloud <https://cozy.io/> ou son YunoHost <https://yunohost.org/>, chacun moins bien protégé que les grosses entreprises professionnelles, mais bénéficiant de la dispersion des données ? Si Facebook est certainement, pour un attaquant, un objectif plus difficile à craquer que le VPS de M. Michu, c'est également un objectif beaucoup plus intéressant : si on le craque, on a les données de centaines de millions de M. Michu. Le RFC suggère donc qu'il vaut mieux ne pas tout mettre chez les GAFA et disperser les objectifs. (Gentiment, il oublie de rappeler que certains attaquants n'ont pas besoin de pirater les GAFA : ils ont PRISM pour cela.)

La session a aussi mis en évidence l'importance d'utiliser un vocabulaire rigoureux quand on parle des menaces et des solutions techniques, surtout quand on s'adresse à un public non-spécialiste. Ainsi, les solutions contre les attaques de l'Homme du Milieu ne « protègent » pas contre ces attaques : elles les rendent simplement visibles. C'est la réaction aux alertes qui déterminera si on est protégé ou pas (pensez au traditionnel avertissement des navigateurs Web : « voulez-vous continuer malgré ce certificat mal fichu ? »).

Une autre session portait sur l'usage des outils de sécurité (section 5.3). Ce n'est pas tout de concevoir des outils géniaux, il faut encore qu'ils soient utilisés. Aujourd'hui, on constate que ce n'est pas assez le cas. Combien de messages circulent en clair quand ils devraient être chiffrés avec PGP ? Combien de SMS envoyés de manière à être lisibles par tous alors qu'il faudrait utiliser Signal (ex-TextSecure <https://www.bortzmeyer.org/textsecure.html>) ? Combien de webmestres renoncent à activer HTTPS parce qu'ils ne veulent pas payer une AC et subir ses procédures et son interface Web mal faite ?

Le RFC note que, dans ce dernier cas, ils ont raison, vu la faille fondamentale de ce système des AC : n'importe quelle AC peut émettre un certificat pour n'importe quel domaine. Donc, il suffit qu'une seule des centaines d'AC reconnues par le logiciel soit compromise pour faire s'effondrer tout l'édifice. Des solutions techniques existent (cf. RFC 6962 ou le RFC 6698, curieusement oublié ici).

Un des problèmes à l'usage des outils de sécurité peut être le modèle de confiance utilisé. Dans X.509, c'est « toutes les AC ont raison ». C'est très dangereux pour la sécurité. PGP a un modèle « réseau de confiance » qui a l'inconvénient de ne bien marcher que pour des petites communautés (il n'est pas réellement transitif). Un modèle longtemps méprisé par les experts en sécurité (ceux avec qui, si on les écoute, on ne déploie jamais aucune solution de sécurité, car aucune n'est parfaite) est le TOFU, utilisé par SSH ou Signal. Il a l'avantage d'être beaucoup plus facile pour les utilisateurs, mais l'attaque de l'Homme du Milieu reste possible pour la première connexion. La technique de l'« épinglage des clés » (RFC 7469) est une variante de TOFU.

Et pour SIP ? La constatation de l'atelier était que les fonctions de cryptographie dans SIP, pourtant bien normalisées, étaient peu utilisées. L'argument des opérateurs SIP est que les mises en œuvre de ce protocole sont peu interopérables et que la seule solution pour que ça marche est de modifier les échanges en cours de route, rajoutant ou enlevant des en-têtes SIP pour s'adapter à chaque logiciel.

(S'ils ont raison, cela signifie que les RFC sur SIP n'étaient pas clairs, ou bien que les programmeurs ne les ont pas lus.)

Peut-être que WebRTC en tirera les leçons (RFC 8827). Après tout, WebRTC est mis en œuvre par les auteurs de navigateurs et ceux-ci n'ont pas les mêmes intérêts que les opérateurs SIP (les intermédiaires adorent modifier les données et détestent la sécurité de bout en bout).

Et pour XMPP (RFC 6120), pourquoi les techniques de sécurité existantes ne sont-elles pas plus fréquemment utilisées ? XMPP a de l'authentification et du chiffrement du canal, grâce à TLS, et de l'authentification et du chiffrement de bout en bout, grâce à OTR (qui a par contre le gros défaut d'être mal intégré au protocole XMPP). Mais le problème identifié pendant la session est plutôt que XMPP est peu utilisé. Contrairement au courrier électronique, où tout le monde utilise les solutions standard (SMTP et IMF), pour la messagerie instantanée, M. Michu fait une confiance aveugle à des systèmes fermés et privatifs comme Skype, dont la sécurité est inconnue ou inexistante.

Il y avait bien sûr une session consacrée aux questions non techniques (section 5.4) car on sait bien maintenant que « la sécurité n'est pas un produit, c'est un processus ». Installer un outil technique ne suffit pas. Il faut aussi traiter les questions d'éducation des utilisateurs, les questions politiques, les questions juridiques... Ainsi, l'excuse du terrorisme est largement utilisée par les États pour justifier des politiques répressives, tellement répressives qu'elles auraient été largement considérées comme inacceptables il y a seulement quelques années (voir par exemple ce qui se fait en France en ce moment avec l'état d'urgence). Il y aussi, note le RFC, le problème des traités internationaux pro-commerce (comme TTIP) qui contiennent souvent des mesures anti-vie privée.

On note que, d'une façon générale, ces problèmes non techniques étaient traditionnellement peu pris en compte par les organisations comme l'IETF, ou par les informaticiens en général. Cela fait très longtemps que les chercheurs en médecine, par exemple, ont des comités d'éthique qui se penchent sur les conséquences de leur travail. Faudrait-il faire la même chose pour les SDO ? Pour l'instant, les normes qui ont reçu une évaluation extérieure sur les questions de vie privée (comme l'API de géolocalisation <http://www.w3.org/TR/geolocation-API/>) l'ont été ponctuellement, sans démarche systématique.

Cela n'interdit pas d'améliorer les outils et une session était consacrée à cette activité (section 5.5 du RFC). D'abord, l'atelier a discuté de la sécurité opportuniste (au sens de « essayer de chiffrer autant que possible, même sans authentification, ce serait mieux que rien »), en considérant qu'elle n'avait pas été assez utilisée jusqu'à présent. L'une des difficultés de cet opportunisme est de ne pas donner de faux sentiments de sécurité à l'utilisateur. Par exemple, il y avait un consensus dans l'atelier que, si un logiciel tente TLS et y réussit, mais sans authentification, il ne doit pas présenter à l'utilisateur des signes rassurants, genre petit cadenas. Le chiffrement ainsi obtenu est un progrès mais il ne faut pas donner l'impression qu'on s'arrête là.

Un aspect délicat à l'heure actuelle est de l'attitude à avoir lorsqu'on ne peut pas chiffrer : est-ce parce que le pair ne sait effectivement pas ou bien parce qu'un Homme du Milieu a réussi à empêcher le chiffrement ? Pour des sessions qu'on veut sécurisées (`https://...` vers sa banque), il faut évidemment s'arrêter là. Mais pour les autres, il serait intéressant, plutôt que de se rabattre sur de la communication en clair, ce qui est souvent le choix, de se souvenir des communications précédentes avec ce pair et de décider en fonction du passé si ce problème de chiffrement est normal (une forme de TOFU).

Là aussi, la terminologie est importante. Les descriptions des connexions utilisant le « chiffrement, si possible » (sécurité opportuniste) parlent souvent d'« échec » si la session chiffrée n'a pu être établie alors qu'un tel échec n'est pas pire que si on n'avait même pas essayé. Il faudrait plutôt parler d'« amélioration » quand

on a réussi à lancer le chiffrement (et ne rien dire si on n'y est pas arrivé). Là aussi, le fait que les concepteurs de protocoles, les experts en sécurité et les programmeurs ne soient pas des spécialistes de l'interface utilisateur va jouer : il n'est pas évident de communiquer des informations sur le niveau de sécurité à l'utilisateur.

Autre point sur lequel il faudrait améliorer les outils, le fait de chiffrer systématiquement, sans action explicite de l'utilisateur. Sinon, il y a un risque que l'utilisateur ne chiffre que les documents sensibles, ce qui facilite la tâche des espions, en leur indiquant quels sont les documents importants. (C'est un des problèmes de PGP, je trouve.)

Il serait également intéressant d'améliorer TOFU. Un des gros problèmes de ce modèle, tel qu'il est utilisé dans SSH, est qu'il n'existe pas de mécanisme propre pour changer la clé d'un serveur (par exemple parce qu'on a mis à jour le logiciel). Idem si on utilise Signal : si votre correspondant a perdu son téléphone et en a acquis un nouveau, il faut manuellement vérifier la nouvelle identité (TOFU ne peut pas savoir si c'est un changement de clé légitime ou bien une attaque de l'Homme du Milieu).

Un sujet qui fait l'objet de nombreuses discussions dès qu'on parle de vie privée et de surveillance est celui des métadonnées (section 5.6). Chiffrer protège en effet très bien le contenu. Malgré quelques rumeurs, il est peu probable que même les plus puissants attaquants arrivent aujourd'hui à craquer le chiffrement bien fait (attention : bien chiffrer est beaucoup plus dur que ce n'en a l'air). Pour un attaquant rationnel, voler les clés, acheter ou menacer un destinataire légitime, ou encore corrompre le logiciel de chiffrement est bien plus aisé que de casser le chiffrement lui-même (c'est d'ailleurs pour cela que les discussions de détail sur la longueur des clés sont assez ridicules). Mais ce chiffrement n'aide pas si les métadonnées sont, elles accessibles. Pour prendre un exemple classique, imaginons deux personnes qui se téléphonent et chiffrent la communication. Le fait qu'ils se téléphonent est lui-même une information, qui peut être suffisante pour un espion, même s'il n'a pas accès au contenu des communications. C'est encore plus vrai si on a accès à plusieurs communications : si, à chaque fois que A appelle B au téléphone, B, immédiatement après avoir raccroché, appelle C, D et E, on en déduira facilement qu'on a identifié une chaîne de commandement, même si on n'a pas le contenu de leurs conversations. Cet exemple illustre bien le danger des métadonnées. Parfois, même la taille des messages donne des indications (si vingt fichiers de taille différente sont sur un serveur, l'observation du trafic, même chiffré, permet de savoir quel fichier a été récupéré).

Or, ces métadonnées sont partout, soit qu'elle soient absolument indispensables au fonctionnement des protocoles (comme l'adresse IP de destination dans les paquets IP), soit que leur potentiel d'indiscrétion n'ait pas été perçu tout de suite. Les supprimer n'est pas évident : un routeur a bien besoin de l'adresse de destination pour router !

Il y a trois techniques pour lutter contre l'espionnage par les métadonnées : l'agrégation, le détour ("*contraflow*"), et la dispersion ("*multipath*"). L'agrégation consiste à regrouper des conversations différentes dans un même flux de données. Par exemple, si votre résolution DNS passe par un gros résolveur partagé, les serveurs qu'interroge ce résolveur auront du mal à identifier vos requêtes, toutes passant par le résolveur. Idem en HTTP si on utilise un relais partagé, sauf évidemment si celui-ci laisse passer, ou, pire, ajoute, des informations discriminantes comme le `User-Agent` : . Même chose côté serveur : si vos pages « sensibles » sont hébergées sur un serveur mutualisé, un observateur aura du mal à dire ce qui était demandé (là encore, tout est dans les détails : par exemple, SNI peut trahir la communication).

Le détour consiste à passer délibérement par des chemins supplémentaires. C'est ce que fait un VPN ou, en encore mieux, Tor. Un espion qui voudrait identifier votre trafic devrait observer l'Internet en de nombreux points, et corréler ce qu'il voit.

Quant à la dispersion, elle consiste à envoyer les données par des moyens différents. Si vous commencez une conversation XMPP via de la 4G et que vous continuez en Wifi, l'attaquant aura davantage de mal que si toute la communication passe par un seul canal.

L'analyse de trafic donne souvent des résultats surprenants d'efficacité, dont les concepteurs de protocole ne se rendent pas compte. (Voir la bibliographie de FreeHaven <http://freehaven.net/anonbib/>, par exemple.) Il existe des méthodes pour la contrarier, comme d'ajouter des messages vides ("*padding*", voir par exemple le RFC 7540, sections 6.1 et 10.7) mais elles ont un coût (par exemple en terme de débit) et ne sont pas faciles à utiliser.

Notez que les protections de bas niveau (couche 3) ne sont pas très utiles si les applications elle-même font fuiter des données. Un navigateur Web est particulièrement terrible sur ce point, voir le Panopticlick <http://panopticlick.eff.org/>. C'est pour cela que Tor recommande l'usage du Tor Browser <https://www.torproject.org/projects/torbrowser.html.en> qui ne se contente pas de router les données via Tor, mais qui s'assure aussi que le navigateur ne vous trahit pas, et n'envoie qu'un minimum de données.

Un problème fréquent pour tous les systèmes de sécurité est celui des "*middleboxes*", ces équipements intermédiaires qui se permettent de filtrer les messages, voire de les modifier. Beaucoup de protocoles de sécurité de l'Internet sont conçus pour une connexion de bout en bout : c'est le cas de TLS, par exemple. La "*middlebox*" viole cette supposition. On voit, par exemple, des "*middleboxes*" qui terminent la session TLS, analysent le trafic, puis re-chiffrent de l'autre côté. Pour passer l'authentification TLS, elles imposent en général de mettre le certificat d'une AC complice sur les ordinateurs clients. À noter que, dans la configuration TLS typique, le client authentifie le serveur mais pas le contraire. Ce qui veut dire qu'un serveur, même ultra-paranoïaque, n'a aucun moyen, contrairement au client, de détecter qu'une "*middlebox*" est présente.

Un autre exemple typique de "*middlebox*" courante est le portail captif (cf. RFC 6585 mais aussi le RFC 7710, issu des réflexions de cet atelier). Certains systèmes d'exploitation utilisent diverses heuristiques pour détecter la présence d'un portail captif afin, par exemple, de permettre une connexion automatisée. Mais ces heuristiques ne marchent évidemment pas toujours. (D'autant plus qu'il semble que certains points d'accès font tout pour empêcher ces heuristiques de marcher, afin d'empêcher les connexions automatisées.)

Idéalement, il faudrait un protocole nouveau permettant aux portails captifs de se signaler, sans avoir à jouer les Hommes du Milieu. Mais, même si un tel protocole était nnormalisé demain, il faut se rappeler que la plupart des points d'accès ne sont jamais mis à jour... (Un hôtel, par exemple, se moque pas mal de si son portail captif marche bien ou pas.)

L'atelier STRINT a ensuite vu les participants se répartir en petits groupes ("*break-outs*") pour discuter en profondeur certains sujets. Par exemple, l'un de ces groupes s'attaquait à l'analyse des clients. Si certains ignorants ne considèrent que les navigateurs Web, les participants à l'atelier savaient bien, eux, qu'il existe une grande variété de logiciels clients. L'un des problèmes de ces clients est « que faire lorsqu'un certificat pose un problème ? » Certains administrateurs système disent simplement aux utilisateurs « continuez, ne vous inquiétez pas de cet avertissement », conseil qu'il sera difficile de faire désapprendre aux utilisateurs ensuite. Peut-être faudrait-il que les logiciels clients ne fournissent plus d'option pour passer outre un problème de certificat. Mais un tel changement devrait être fait plus ou moins simultanément par tous les auteurs de logiciels de cette catégorie. Autrement, il y a un risque que l'utilisateur croit que les logiciels les plus sûrs sont en tort, en l'empêchant de se connecter à son serveur.

Autre problème d'interaction entre la sécurité informatique et les facteurs humains, le cas des certificats auto-signés. Si un tel certificat est évidemment inacceptable pour une banque ou pour GitHub,

pour un site personnel, il peut être tout à fait valable et en tout cas meilleur qu'une communication en clair. Un des aspects délirants de l'usage de HTTPS aujourd'hui est que bien des gérants de serveurs personnels n'activent pas HTTPS car obtenir un certificat est cher et compliqué, et utiliser un certificat auto-signé provoquerait des avertissements de sécurité injustifiés. Résultat : on reste en clair...

Un autre groupe travaillait sur l'activation par défaut des meilleurs choix de sécurité. Quant l'IETF dit que tous les programmes DOIVENT ("*MUST*", cf. RFC 2119) mettre en œuvre tel protocole ou tel algorithme, cela veut dire qu'il doit être présent dans le code (MTI ou "*Mandatory To Implement*", cf. RFC 3365). Mais cela ne signifie pas qu'il sera utilisé, s'il faut une action explicite de l'utilisateur ou de l'administrateur système pour l'activer. La très grande majorité des utilisateurs, et la plupart des administrateurs système, ne changent pas les réglages par défaut. Si on veut vraiment combattre la surveillance généralisée, il ne suffit pas d'avoir 2 ou 3 % des utilisateurs qui sont protégés. Il faut que presque tous le soient, donc passer du MTI au MTU ("*Mandatory To Use*").

Enfin, un autre groupe "*break-out*" a planché sur la notion de « sécurité opportuniste » ("*opportunistic security*"). Le terme est très utilisé dans les milieux de la sécurité informatique mais n'a pas de définition précise. Plusieurs auteurs ont essayé d'en donner une définition (voir par exemple le RFC 4322) mais ces définitions sont très différentes les unes des autres <http://www.ietf.org/mail-archive/web/ietf-privacy/current/msg00337.html>. Depuis, l'IETF a publié la sienne, dans le RFC 7435.

On l'a dit, ce RFC arrive bien tard après l'atelier. Depuis, du travail a été fait. Juste après l'atelier, la réunion IETF de Londres <http://www.ietf.org/meeting/89/index.html> a logiquement beaucoup couvert ces questions de protection contre l'espionnage. Ce fut par exemple la « "*DNSE BoF*" », première réunion physique du projet « DNS et vie privée », qui a depuis débouché sur le RFC 7626.

La totalité des articles présentés est en ligne <https://www.w3.org/2014/strint/report.html#papers>.