Des clés dans le DNS, un successeur à X.509 ?

Stéphane Bortzmeyer
AFNIC, service R&D
Immeuble International
78181 Saint-Quentin-en-Yvelines

Résumé

Après plusieurs années pendants lesquelles les certificats numériques à la norme UIT X.509 étaient présentés comme la solution ultime à tous les problèmes de communication sécurisée sur l'Internet, l'inquiétude est grande. D'une part, on constate une prise de conscience des limites de ce système : une Autorité de Certification peut faire ce qu'elle veut, elle n'est pas contrôlée, et peut émettre des certificats pour des entités qui ne sont pas ses clients. D'autre part, la simple concurrence commerciale tend à faire baisser toujours plus la qualité des certificats. Mais, en 2011, le problème a soudain changé d'échelle avec le piratage des AC Comodo et Diginotar. Tout à coup, tout le monde se rendait compte du sérieux risque que courrait tout utilisateur de l'Internet si une seule AC est piratée ou simplement malveillante.

C'est désormais le modèle de base de X.509 qui est mis en question. Pour le remplacer, plusieurs pistes sont suivies, la principale étant celle qui consiste à mettre les certificats, ou bien, dans une version plus modérée, des méta-données sur les certificats, dans le DNS. Celui-ci disposant désormais d'un mécanisme de résolution sécurisée, avec DNSSEC, peut-il remplacer X.509 ?

Exposé aux Journées Réseaux (JRES) à Toulouse le 23 novembre 2011.

Mots clefs

DNS, DNSSEC, X.509, sécurité

1 Introduction

Il y a bien longtemps que le modèle de sécurité de X.509, ses certificats et son IGC (Infrastructure de Gestion de Clés), est critiqué, alors même que les autorités et les vendeurs font comme si de rien n'était, comme si ce mécanisme était digne de porter l'essentiel de la sécurité de l'Internet. Ellison et Schneier avaient ouvert le feu en 2000 avec « *Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure* »[1]. L'exposé de Serge Aumont « Faut-il brûler vos certificats ? » à JRES 2003 avait été un des premiers textes en français à pointer les limites de cette technologie. Depuis, elle a à nouveau souvent été sous le feu des critiques. En mai 2010, la décision de Mozilla d'intégrer le certificat de l'Autorité de Certification CNNIC, émanation de la dictature chinoise, avait suscité une série de protestations [2], certaines parfois à la limite du racisme (CNNIC était une des premières AC non-occidentales). Mais l'affaire avait également révélé l'une des faiblesses les plus sérieuses de X.509 : son absence de limitation des pouvoirs de l'AC. Toute AC peut créer un certificat pour tout nom et il n'y a pas de moyen, par exemple, de limiter CNNIC à des noms en .CN.

Cette faiblesse a été activement exploitée par des entreprises qui souhaitaient espionner leurs employés (et qui installent le certificat d'une AC « *corporate* » sur tous les postes, pour pouvoir intercepter même les flux chiffrés, ou par des gouvernements comme l'ancienne dictature tunisienne qui détournait les communications Gmail ou Facebook, même sécurisées en HTTPS, en profitant de certificats gouvernementaux, intégrés dans certains navigateurs [3].

Il s'agissait là d'usages « normaux » de X.509, conformes à son modèle de sécurité (mais pas aux attentes des utilisateurs). En mars 2011, la révélation du fait que l'AC Comodo laissait un de ses revendeurs produire des certificats frauduleux [4] pour des sites sensibles (comme Gmail ou comme addons.mozilla.org, qui sert pour les mises à jour logicielles automatiques de Firefox) a encore augmenté l'inquiétude. Couplée avec des attaques sur l'infrastructure (par exemple DNS ou BGP), ces vrais-faux certificats permettaient à l'attaquant de se faire passer pour ces sites sensibles.

Encore pire, à l'été 2011, il est apparu que l'AC Diginotar[5] avait été piratée. Cette fois, les vrais-faux certificats avaient été générés sans passer par le système normal, ce qui fait qu'il n'y avait aucune trace de leur existence[1]. Aucun moyen donc d'émettre des révocations signées. L'excellent rapport de Fox[6] notait qu'on pouvait trouver l'utilisateur des certificats par les requêtes des logiciels cherchant à vérifier si ces révocations avaient été faites[2] : la plupart des requêtes venaient d'Iran, ce qui indique que le gouvernement iranien espionnait ses citoyens, notamment lorsqu'ils utilisaient Gmail (le premier vrai-faux certificat repéré était pour les services de Google).

Si cette attaque était très spectaculaire, des problèmes de plus basse intensité sont très fréquents : le prix toujours plus bas des certificats ne permet pas de sérieusement espérer que des vérifications dignes de ce nom soient faites.

Existe-il aujourd'hui des alternatives à X.509, qui pourraient remplacer cette technique invalide, qui ne tient que par l'affirmation répétée qu'elle sécurise l'Internet, en dépit des preuves du contraire ? Il en existe trois catégories, qui ne sont d'ailleurs pas forcément incompatibles. L'avenir résidera peut-être dans une combinaison des trois. Je vais d'abord détailler les problèmes de X.509 avant d'aborder chacune à son tour ces trois solutions.

2 Le fonctionnement de X.509

X.509 est une norme UIT très complexe, qui couvre un grand nombre des aspects du fonctionnement d'une IGC. Cette norme définit un format de données, une sémantique des certificats, des pratiques de validation, etc. Le terme « certificat », dans ce sens, désigne une clé publique cryptographique, avec des méta-données (notamment des dates de début et de fin de validité), et surtout la signature d'une AC (Autorité de Certification) qui garantit que le titulaire de la clé privée associée (en termes X.509, il se nomme le sujet) est bien celui qu'il prétend être. Comment les AC sont-elles choisies ? En pratique, ce sont les éditeurs de logiciels qui décident des AC[3] que reconnaîtra leur mise en œuvre de X.509. L'éditeur d'un navigateur Web a donc un pouvoir important. L'utilisateur a la possibilité de modifier cette liste d'AC mais seul un nombre infime le fait. On voit aussi dans la nature bien des certificats auto-signés (ou, dit autrement, où l'AC est l'utilisateur) qui ne valent pas moins que les autres : il n'existe pas de bons ou de mauvais certificats, tout dépend de la confiance qu'on accorde à l'émetteur.

Voici par exemple le certificat d'une AC, Equifax, affiché avec l'outil en ligne de commandes d'OpenSSL. Equifax le signe elle-même, étant à la racine de la validation :

```
% openssl x509 -text -in ./Equifax.pem
Certificate:
 ....
     Issuer: C=US, O=Equifax Secure Inc., CN=Equifax Secure Global eBusiness CA-1
     Validity
        Not Before: Jun 21 04:00:00 1999 GMT
        Not After : Jun 21 04:00:00 2020 GMT
     Subject: C=US, O=Equifax Secure Inc., CN=Equifax Secure Global eBusiness CA-1
     ...
     X509v3 extensions:
...
```

Et celui d'un site Web, manager.linode.com, signé par la même AC. Celle-ci garantit donc que la machine qui peut signer des messages avec cette clé est bien manager.linode.com. On utilise un autre logiciel[4] pour afficher le certificat. Notez que cet autre logiciel a un magasin de certificats vide et ne connait donc pas Equifax:

```
% gnutls-cli -p 443 manager.linode.com
```

[1] Normalement, l'AC enregistre tout ce qu'elle fait, pour permettre un audit ultérieur.

[2] Ce n'est qu'une heuristique : un attaquant plus sophistiqué aurait pu également bloquer le protocole OCSP qui sert à cette vérification.

[3] Certains éditeurs, comme Mozilla, publient leur politique d'acceptation des AC http://www.mozilla.org/projects/security/certs/policy/InclusionPolicy.html Voir aussi http://isc.sans.edu/diary.html?storyid=11530 En France, ceux qui s'intéressent à cette question peuvent aussi regarder les obligations du RGS [7] et le décret n°2001-272 du 30 mars 2001, NOR: JUSC0120141D, Chapitre III, Article 6, II, m.

[4] Inclus dans l'excellente mise en œuvre de TLS, GnuTLS.

```
...
 - Got a certificate list of 1 certificates.
 - Certificate[0] info:
  - subject
`serialNumber=BPfRE/o7Tg0NKM5Zqkno0qjNCJRy/2Gz,C=US,O=*.linode.com,OU=GT15858513,OU=See
www.rapidssl.com/resources/cps (c)10,OU=Domain Control Validated - RapidSSL(R),CN=*.linode.com', issuer
`C=US,O=Equifax,OU=Equifax Secure Certificate Authority', RSA key 1024 bits, signed using RSA-SHA1,
activated `2010-05-03 19:39:54 UTC', expires `2013-07-04 22:58:12 UTC', SHA-1 fingerprint
`a7e20f15e4e3cd0653c90636528ea96999ce8c1d'
 - The hostname in the certificate matches 'manager.linode.com'.
 - Peer's certificate issuer is unknown
 - Peer's certificate is NOT trusted
```

3 Les usages de X.509

À quoi sert X.509 en pratique ? Une des utilisations la plus connue est de protéger TLS. Ce protocole, normalisé dans le RFC 5246 [8], permet de chiffrer une session avec une machine distante, empêchant ainsi l'écoute par un indiscret. TLS est à son tour utilisé dans de nombreux protocoles Internet, le plus connu étant HTTPS. Pourquoi est-ce que TLS a besoin de X.509, même si on n'authentifie pas, ou même si l'authentification se fait par un autre moyen[5] ? Parce que TLS permet de faire face à deux sortes d'attaquants : les passifs, qui se contentent d'écouter (X.509 est alors inutile, on peut utiliser TLS avec n'importe quelle clé, même non signée) et les actifs qui peuvent modifier les paquets en route. Dans ce dernier cas, TLS seul ne suffit pas car on ne peut pas être sûr qu'on parle bien avec la machine visée : celle qui a répondu peut être contrôlée par l'attaquant[6]. Seul un certificat signé permet de s'assurer que le redoutable Homme du Milieu ne s'est pas glissé dans la conversation.

L'usage le plus courant de TLS est donc que le serveur[7] envoie un certificat signé. Le client vérifie que cette signature a été apposée par une des AC[8] qui figurent dans son magasin de certificats, et que le nom de la machine correspond [9]. Ledit magasin est typiquement pré-rempli par l'éditeur du logiciel et très peu d'utilisateurs regardent ce qu'il y a dedans.

Notez que les attaques de l'Homme du Milieu ne sont pas rares : on trouve ainsi des entreprises qui mettent un relais transparent sur le port 443, envoient le trafic HTTPS à un serveur qui termine les sessions TLS, puis les relaie vers le vrai serveur. Cela nécessite l'installation du certificat de l'entreprise sur tous les postes de travail mais, une fois que c'est fait, c'est un puissant outil d'espionnage.

4 Les limites et problèmes de X.509

X.509 est en production sur l'Internet depuis de nombreuses années (et dans d'autres contextes depuis encore plus longtemps). Les AC existantes, et leurs utilisateurs, ont beaucoup d'expérience. Cette expérience a permis de détecter trois gros problèmes :

1. Les utilisateurs ne comprennent pas toujours ce que garantit et ne garantit pas X.509. Par exemple, l'AC ne promet qu'une authentification, pas une évaluation du sérieux ou du caractère honnête de l'organisation qui est derrière un site Web. Des millions d'utilisateurs ont été entraînés à accepter une page Web « s'il y a le petit cadenas en bas » (signe que la session est chiffrée avec TLS) sans comprendre que TLS et X.509 ne signifiaient pas que le site de e-commerce visité était digne de confiance.

2. Les AC sont presque toutes des entreprises privées à but lucratif [10]. Les incitations économiques ne vont pas dans le sens de plus grandes vérifications (songez au coût, pour une AC située aux États-Unis, de vérifier que telle entreprise au

[5] Mot de passe, par exemple.

[6] L'attaquant a deux grandes catégories de méthodes pour se mettre ainsi au milieu de la communication : soit il contrôle déjà l'infrastructure, par exemple parce qu'il est le fournisseur d'accès à l'Internet, soit il arrive à pirater des protocoles comme le DNS ou comme BGP, redirigeant ainsi la victime vers le serveur de son choix.

[7] Dans le contexte de l'Internet, on authentifie rarement le client, le serveur acceptant tous les clients.

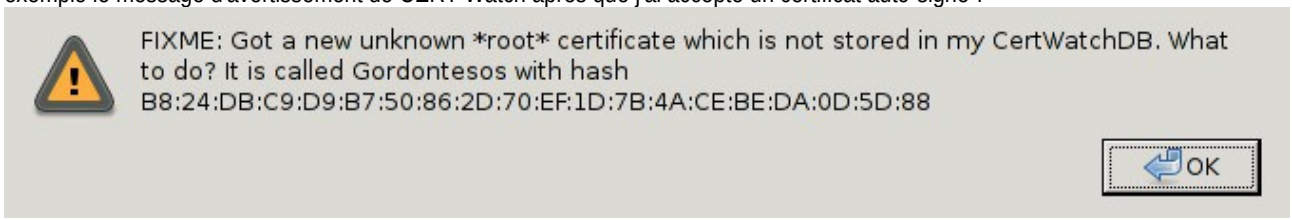
[8] X.509 offre bien d'autres possibilités, notamment de chaînes de confiance, où l'AC signe le certificat d'une organisation qui va à son tour signer un autre certificat, etc, jusqu'à celui du serveur. Il y a donc une différence entre racine de certification, le point de départ de la validation, et autorité de certification.

Pakistan existe vraiment...) Cela mène à une course au moins cher et les prix des certificats ne permettent pas d'imaginer des vérifications sérieuses. Résultat, on voit parfois apparaître des « super-certificats » comme EV actuellement, qui promettent, pour beaucoup plus cher, de faire ce qui était déjà promis avant...

3. Du point de vue technique, la plus grosse faiblesse de X.509 est qu'un certificat de n'importe quelle AC du magasin est accepté, sans savoir si le sujet authentifié est client ou pas de cette AC (information qui n'apparait pas dans X.509). Il y a entre 500 et 700 AC dans le magasin d'un navigateur Web typique et il suffit à l'attaquant d'en pirater une seule pour pouvoir se faire passer pour n'importe qui. Ainsi, il ne sert à rien de choisir une AC « sérieuse » et chère, ce qui compte n'est pas la sécurité de l'AC dont on est client, mais la sécurité de la plus mauvaise AC du magasin.

5 TOFU

La première alternative à l'IGC est le modèle TOFU (*Trust On First Use*) qui est surtout connu par son utilisation dans SSH. En gros, l'idée est d'enregistrer la clé du pair à la première connexion (qui est donc vulnérable) et de surveiller ensuite les changements, indicateurs d'une attaque. Ce modèle est par exemple promu par Chris Palmer dans son fameux exposé « *It's Time to Fix HTTPS* »[11]. Il peut être amélioré en échangeant de l'information entre pairs de confiance (voir l'alternative suivante). On peut noter que cette idée de surveillance des changements est déjà mise en œuvre dans des solutions X.509 comme CERT Patrol[9] ou CERT Watch[10], qui permettent de détecter des attaques comme celle résultant des vrais-faux certificats de Comodo ou Diginotar. Voici par exemple le message d'avertissement de CERT Watch après que j'ai accepté un certificat auto-signé :



Pour des exemples avec CERT Patrol, voir [12].

Pour plus de détails, prenons l'exemple de SSH, le plus classique : dans la configuration par défaut d'OpenSSH[11], la première fois qu'on se connecte à une nouvelle machine, le client note que la machine est inconnue et demande confirmation à l'utilisateur. Celui-ci peut vérifier la clé affichée (personne ne le fait...) et, si elle est acceptée, elle est enregistrée et sera reconnue les fois suivantes. Le gros avantage du modèle TOFU est sa simplicité. Alors que tant de systèmes de cryptographie n'ont jamais connu aucun déploiement réel, car ils embêtaient trop l'utilisateur pour être acceptés, SSH a été massivement déployé en très peu de temps.

Sa principale faiblesse, comme son nom l'indique, est que la première connexion n'est pas réellement protégée (sauf pour ceux qui vérifient la clé la première fois). Il faut donc prier que l'attaquant, le fameux Homme du Milieu, ne soit pas sur le chemin de la connexion la première fois qu'on fait du SSH avec une nouvelle machine.

Une autre faiblesse de ce modèle est la révocation : lorsqu'on change une clé, l'utilisateur va être prévenu d'un problème (c'est ainsi que Cert Watch ou Cert Patrol génèrent souvent des avertissements inutiles) et il devra accepter ce changement.

6 Réseau de confiance

La deuxième catégorie de solution alternative est celle qui repose sur un réseau de confiance, comme dans PGP. On peut imaginer des signatures des clés, effectuées par des pairs, et que la confiance diminue petit à petit lorsqu'on s'éloigne (j'ai plus confiance dans mes amis que dans les amis de mes amis). C'est un tel système qu'essaient de développer les projets comme Monkeysphere[13] ou Convergence [14]. L'avantage de telles méthodes est qu'elles sont intégralement pair-à-pair, sans composant central.

---

[9] https://addons.mozilla.org/en-US/firefox/addon/certificate-patrol/

[10] https://addons.mozilla.org/en-US/firefox/addon/certificate-watch/

[11] On peut choisir une configuration plus paraoïaque, où la clé de la machine distante devra être mise explicitement dans la configuration locale, sur Unix le fichier ~/.ssh/known_hosts.

Ces méthodes sont compatibles avec le TOFU, et peuvent donc l'améliorer : on accepte les clés connues, et, lorsqu'elles ne le sont pas, on demande à un réseau de pairs s'ils ont déjà vu passer cette clé. C'est ce que fait le système Perspectives pour SSH[15].

7 Les clés dans le DNS

La troisième catégorie de solution de remplacement pour X.509 repose sur une observation simple : la plupart des interactions sur l'Internet commencent par un nom de domaine et donc par une résolution DNS [12] [13]. Pourquoi donc développer une nouvelle infrastructure de confiance, l'IGC, lorsqu'on a celle des noms de domaine, dont on est déjà dépendant ? Bien sûr, cela suppose que la résolution DNS soit sécurisée, et donc cette idée, pourtant ancienne [16] [17] n'a réellement décollé que très récemment, avec le déploiement important de DNSSEC.

Cette idée a donc poussé à la création d'un groupe de travail IETF[14]. Nommé d'abord KIDNS (pour « *Keys In DNS* »), il se nomme désormais DANE (pour « *DNS-based Authentication of Named Entities* ») et a pour tâche de décrire un nouveau mécanisme pour stocker des clés dans le DNS, sans validation extérieure. Ainsi, le titulaire de www.example.org pourra sécuriser https://www.example.org/ sans recourir à des tiers à la fiabilité douteuse, uniquement en mettant les clés dans la zone DNS qu'il contrôle. Conséquence de la nature arborescente du DNS, le titulaire de example.org n'aura donc aucune possibilité de tricher sur example.com. CNNIC (registre de .CN et AC) pourrait certes tricher sur les noms en *.cn mais uniquement sur ceux-ci, contrairement à la situation actuelle avec X.509.

Le projet DANE est donc loin d'être terminé et toute description est donc forcément temporaire. En octobre 2011, le seul texte terminé par le groupe de travail était l'exposé des usages, le RFC 6394 [18]. Ce document explique trois scénarios d'usage typiques de DANE, en partant des classiques Alice (qui gère le serveur TLS, alice.example.com), Bob (son client) et Charlie (l'AC sérieuse et connue) :

1. Le premier scénario est le cas où Alice a un certificat délivré par Charlie, en est très contente mais craint que DigiNotar ou bien une autre AC méchante ou piratée, fasse un faux certificat pour son site. Elle souhaite utiliser DANE pour dire dans le DNS, « Mon AC, c'est Charlie et personne d'autre », fermant ainsi la principale vulnérabilité de X.509. On nomme cela la « contrainte sur l'AC » (type 0 en jargon DANE).

2. Dans le deuxième scénario, Alice a un certificat de Charlie et craint que Charlie ne soit pas si digne de confiance que cela et émette des certificats alice.example.com pour des méchants. Elle voudrait pouvoir publier dans le DNS l'identité de son certificat (son condensat, par exemple). Dans ces deux premiers cas, on voit que le DNS ne vient qu'en supplément de X.509 (et peut donc se contenter d'une faible sécurité). Cela se nomme « contrainte sur le certificat » (dit aussi type 1).

3. Le troisième scénario est celui où Alice n'a plus du tout confiance dans les AC et voudrait publier elle-même son propre certificat. C'est le scénario « maximal » pour DANE, celui d'un remplacement complet de X.509, qui fait donc porter une grosse responsabilité sur les opérateurs DNS. Ce sont les « certificats émis par le domaine », alias le type 2 de DANE.

Comment fonctionne DANE ? Cela sera normalisé dans un futur RFC, actuellement pas terminé [19]. Le principe est de mettre dans le DNS un enregistrement d'un type nouveau, TLSA, qui permet d'associer à un nom le certificat à utiliser. Le nom de l'enregistrement sera celui du serveur TLS, préfixé du protocole et du numéro de port. La valeur de l'enregistrement indiquera la sémantique du certificat (type 0, type 1 ou type 2) et le certificat lui-même. Par exemple,

```
_443._tcp.www.example.com. IN TLSA (
   0 0 1 d2abde240d7cd3ee6b4b28c54df034b9
      7983a1d16e8a410e4561cb106618e971 )
```

est un enregistrement pour une connexion TCP au port 443 de la machine www.example.com (probablement pour du HTTPS). Cet enregistrement est de type 0 (le premier champ). Il indique donc qu'on renvoie au certificat de l'AC utilisée (on ajoute donc une contrainte au fonctionnement normal de X.509, cette contrainte évite les attaques type DigiNotar, où une AC émet un certificat pour une entreprise qui n'est pas son client). Au cours de la négociation TLS, le client exigera un certificat qui se rattache à cette AC. Le

[12] *Domain Name System*

[13] Contrairement à ce qu'on lit souvent, c'est aussi le cas si on utilise un moteur de recherches – il faut bien trouver l'adresse IP du moteur, et celui-ci renvoie des URL pas des adresses – ou bien certains protocoles pair-à-pair – le torrent de BitTorrent se trouve en général via un URL.

[14] *Internet Engineering Task Force*, principale organisation de normalisation sur l'Internet. http://www.ietf.org/

deuxième champ, également à zéro, identifie cet enregistrement comme stockant un certificat complet, pas juste la clé. Et le troisième champ, qui vaut 1 , indique qu 'on ne stocke pas le certificat mais un condensat SHA-256 de celui-ci (le quatrième champ).

Cet autre enregistrement :

```
_443._tcp.www.example.com. IN TLSA (
    2 0 0 30820307308201efa003020102020... )
```

est de type 2 : l'IGC X.509 n'est plus du tout utilisée, le certificat stocké dans le DNS suffira au client DNS. Le troisième champ, mis à 0, indique que c'est le certificat complet, pas juste son condensat, qui est mis dans le DNS. Les certificats pouvant être de grande taille, il faudra s 'assurer qu'il n'y pas devant le serveur de noms de pare-feu configuré par un incompétent et qui, par exemple, limiterait les paquets DNS à 512 octets (ce qu'on voit encore sur les sites les plus mal gérés).

Les enregistrements TLSA sont des enregistrements DNS normaux. Ils ont donc une durée de vie (le champ TTL du DNS), et c'est celle-ci qui permet leur remplacement (il n'y a pas de révocation explicite).

Et que doit faire le client TLS si quelque chose ne se passe pas normalement lors de la vérification DANE ? D'abord, et avant tout, bien que les futurs RFC donnent des indications sur les comportements souhaitables, c'est une question de politique locale, comme pour un résolveur validant avec DNSSEC, ou un client SSH qui a été configuré pour être plus ou moins paranoïaque lorsqu'une clé de serveur ne correspond pas à ce qu'il attend. Le client TLS peut donc décider de se rabattre sur du X.509 habituel, ou au contraire imposer la présence d'un enregistrement TLSA valide.

À l'heure actuelle, le type TLSA n'est pas encore connu des logiciels DNS comme dig. On trouve donc sur le réseau des enregistrements expérimentaux (TYPE257...) uniquement, et que dig ne sait pas formater correctement[15].

Il n'existe aujourd'hui pas de mise en œuvre de DANE largement disponible. La seule qui ait été distribuée est une version récente du navigateur Web Google Chrome[16]. Il faut dire que le travail de normalisation n'est pas fini et que, comme indiqué plus haut, même le numéro du type d'enregistrement TLSA n'est pas fixé. On peut quand même être optimiste : les développeurs de deux éditeurs de navigateurs Web (Google et Mozilla) participent activement au groupe de travail DANE et ont indiqué leur intention d'en doter leur logiciel.

8 Sécurité du DNS

Arrivé à ce stade, nous sommes apparemment proches d'une solution. Nous publions des enregistrements DNS indiquant (directement ou indirectement) le certificat ou l'AC utilisé et nous sommes en sécurité. Mais, pour que cela soit vrai, il faudrait que le DNS lui-même soit sûr. Or, ce n'est pas le cas. La quasi-totalité des requêtes DNS passent sur UDP, un protocole qui n'a aucune protection contre l'usurpation (il est trivial de faire une fausse réponse UDP) et la seule protection du DNS est un champ « *Query ID* » qui ne fait que 16 bits, ce qui est très insuffisant[17]. Cette faiblesse du DNS est connue depuis de nombreuses années mais a été brusquement mise sur le devant de la scène en 2008 par la publication de l'attaque Kaminsky[20]. Cette attaque permet d'empoisonner un résolveur DNS avec de fausses informations en quelques minutes. Des changements dans les serveurs ont permis de repousser le problème temporairement mais ces changements[18] ne résisteront pas aux progrès des réseaux.

D'où le déploiement de DNSSEC [21], qui a effectivement commencé en 2007, avec la signature de .SE, le premier domaine de tête à être signé. DNSSEC permet de signer cryptographiquement les enregistrements DNS , garantissant ainsi au résolveur DNS la possibilité de valider les informations obtenues. Son déploiement, qui était longtemps resté du domaine du vœu, est désormais très avancé. La racine est signée depuis juillet 2010, .FR en septembre de la même année et .COM en mars de l'année suivante. Tous les domaines de tête importants le sont également. En revanche, pour l'instant, les domaines de deuxième ou troisième niveau sont rarement signés, et très peu de résolveurs valident les signatures[19].

---

[15] Vous pouvez regarder celui de dnssec.imperialviolet.org par exemple.

[16] http://www.imperialviolet.org/2011/09/19/dnsseclive.html

[17] C'est l'équivalent d'un mot de passe qui ferait entre deux et trois caractères...

[18] Port source UDP aléatoire

[19] Signalons en France ceux de Lothaire, la plaque régionale Lorraine du NREN.http://reseau.ciril.fr/wikidoc/Services/DNS

DNSSEC permet de résoudre le problème mentionné plus haut : si les enregistrements DANE (de type TLSA) sont signés avec DNSSEC, il n'y a plus de risque à les utiliser. Bien qu'il y ait une discussion en cours en ce moment à l'IETF sur la possibilité d'avoir des enregistrements DANE non-signés (notamment pour ceux de type 0 et de type 1, qui ne fonctionnent qu'en supplément de l'IGC classique), il est probable que le déploiement de DANE suivra celui de DNSSEC. Donc, pour simplifier, posons le principe : on ne déploie DANE que lorsque DNSSEC est déjà en place.

Cela peut laisser l'application (par exemple le navigateur Web) dans l'ignorance. En effet, les API existantes ne donnent pas en général de moyen simple de savoir si l'enregistrement DNS a été validé et par qui. Une application cliente TLS qui obtient un certificat via DANE peut-elle lui faire confiance ou pas ? À l'heure actuelle, où la grande majorité des enregistrements DNS n'est pas signée, on ne peut pas encore configurer les résolveurs pour qu'ils exigent cette signature. Il est donc crucial que des mécanismes soient développés pour porter l'information sur la signature à la connaissance de l'application.

Une question liée à celle-ci est celle du « dernier kilomètre ». Lorsque la validation est faite par un résolveur lointain, pas forcément digne de confiance, elle n'a pas la même valeur que lorsqu'elle est faite par un programme tournant sur la machine locale. Même si le résolveur lui-même est fiable, le « dernier kilomètre » n'est pas protégé par DNSSEC et des attaques y sont donc possibles. Il est donc probable que le futur verra le déploiement de résolveurs validants sur les postes de travail (la sécurité doit se faire de bout en bout), comme le permet le logiciel dnssec-trigger [22].

Une autre inquiétude liée au système DANE [23] [24]provient de la confiance (ou du manque de confiance) dans la chaîne d'acteurs qui permet de placer et de modifier des enregistrements dans le DNS. Dans le cas le plus courant, la sécurité de cette chaîne, entre la banque qui gère mabanqueamoi.fr et les serveurs de noms de .FR, dépend de plusieurs acteurs : la banque, son bureau d'enregistrement, le registre, l'hébergeur DNS[20]. Si l'un d'entre eux est défaillant, ou malhonnête, tout l'édifice est ébranlé. Et il n'y a pas de raison de penser que les bureaux d'enregistrement ou les registres soient « meilleurs » ou « plus honnêtes » que les AC.

En analysant ce risque, il faut d'abord se rappeler, qu'avec beaucoup d'AC actuelles, les seules vérifications effectivement faites sont la capacité à recevoir du courrier électronique et/ou à modifier une page Web sous le nom demandé. Autrement dit, aujourd'hui, si on contrôle le DNS de mondomaine.example, on peut déjà obtenir un certificat pour mondomaine.example (rappelez-vous que, avec X.509, l'attaquant n'a pas besoin d'obtenir un certificat de la même AC que la victime).

Mais, surtout, il faut se dire qu'une organisation qui est présente sur l'Internet fait déjà confiance, à tort ou à raison, à ces acteurs. L'avantage de DANE est d'éviter de multiplier les acteurs, et de réutiliser une chaîne de confiance qui existe déjà.

Pour reprendre le mot de Paul Wouters, « *DANE is not the silver bullet, but the CA system is shooting soap bubbles.* »

9 Conclusion

Aujourd'hui, DANE n'est pas terminé : les RFC ne sont pas sortis, il y a peu de mises en œuvre du protocole, et un préliminaire nécessaire, DNSSEC, n'est pas encore partout. On n'a donc pas de solution immédiate pour améliorer ou remplacer le système très peu satisfaisant des IGC. Mais l'IETF y travaille activement, deux éditeurs de navigateurs Web participent au projet, et il est probable que normes et logiciels apparaîtront début 2012.

Le reste est du marketing : les utilisateurs seront-ils convaincus ? Préférerons-t-ils la nouvelle technique plus sûre ou bien l'ancienne, dont les failles sont connues, et dont l'ancienneté rassure ?

10 Bibliographie

[20] Qui est parfois le bureau d'enregistrement, parfois la banque et parfois un tiers.

1. C. Ellison and B. Schneier « Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure » <http://www.schneier.com/paper-pki.html>
2. Marsh Ray « Has Mozilla lost their … oh never mind » <http://extendedsubset.com/?p=33>
3. Seth Schoen « New Research Suggests That Governments May Fake SSL Certificates » <https://www.eff.org/deeplinks/2010/03/researchers-reveal-likelihood-governments-fake-ssl>
4. Comodo « Fraud Incident » <http://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html>

5. Lance Whitney « Comodohacker returns in DigiNotar incident » 2011 <http://news.cnet.com/8301-1009_3-20102027-83/comodohacker-returns-in-diginotar-incident/>

6. Fox-IT BV « DigiNotar Certificate Authority breach / "Operation Black Tulip" » <http://www.rijksoverheid.nl/ministeries/bzk/documenten-en-publicaties/rapporten/2011/09/05/diginotar-public-report-version-1.html>

7. Gouvernement français « Référentiel général de sécurité » <http://www.ssi.gouv.fr/fr/reglementation-ssi/referentiel-general-de-securite/>

8. T. Dierks & E. Rescorla « RFC 5246 The Transport Layer Security (TLS) Protocol Version 1.2 » <http://www.rfc-editor.org/rfc/rfc5246.txt>

9. P. Saint-Andre & J. Hodges « RFC 6125 Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS) » <http://www.rfc-editor.org/rfc/rfc6125.txt>

10. Newsoft « SSL est cassé » <http://news0ft.blogspot.com/2010/04/ssl-est-casse.html>

11. Chris Palmer « It's Time to Fix HTTPS » <https://docs.google.com/present/view?id=df9sn445_206ff3kn9gs&pli=1>

12. Seb Sauvage « Je suis content d'utiliser CertPatrol » <http://sebsauvage.net/rhaa/index.php?2010/11/08/19/54/41-je-suis-content-d-utiliser-certpatrol>

13. « The Monkeysphere Project » <http://web.monkeysphere.info/>

14. Gordon Tesos « Brainstorming : Hacking SSL » <https://gordon.so/internet/brainstorming-hacking-ssl.html>

15. Stéphane Bortzmeyer « Perspectives, un outil pour améliorer la sécurité de SSH et des protocoles équivalents » <http://www.bortzmeyer.org/perspectives-ssh.html>

16. J. Schlyter & W. Griffin « RFC 4255 Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints » <http://www.rfc-editor.org/rfc/rfc4255.txt>

17. S. Josefsson « RFC 4398 Storing Certificates in the Domain Name System (DNS) » <http://www.rfc-editor.org/rfc/rfc4398.txt>

18. R. Barnes « RFC 6394 Use Cases and Requirements for DNS-Based Authentication of Named Entities (DANE) » 2011 <http://www.rfc-editor.org/rfc/rfc6394.txt>

19. P. Hoffman & J. Schlyter « Using Secure DNS to Associate Certificates with Domain Names For TLS » <http://tools.ietf.org/id/draft-ietf-dane-protocol>

20. Stéphane Bortzmeyer « Comment fonctionne la faille DNS « Kaminsky » ? » <http://www.bortzmeyer.org/comment-fonctionne-la-faille-kaminsky.html>

21. Stéphane Bortzmeyer « Sécurité du DNS et DNSSEC » 2009 <https://2009.jres.org/planning_files/article/pdf/5.pdf>

22. Stéphane Bortzmeyer « dnssec-trigger, un outil pour mettre DNSSEC à la disposition de M. Toutlemonde » <http://www.bortzmeyer.org/dnssec-trigger.html>

23. Lauren Weinstein « DNS + DANE = Dumb, Dumber, Disaster - Or - How to Wreck Secure Internet Communications » <http://lauren.vortex.com/archive/000873.html>

24. Moxie Marlin Spike « SSL And The Future Of Authenticity » <http://blog.thoughtcrime.org/ssl-and-the-future-of-authenticity>