Plusieurs noms dans un certificat X.509

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 14 novembre 2008

http://www.bortzmeyer.org/plusieurs-noms-dans-certificat.html

---

X.509 est l'énorme usine à gaz qui permet au secrétaire général de l'ITU de se vanter <https://cai.icann.org/files/meetings/cairo2008/toure-speech-06nov08.txt> de la contribution de son organisation à l'Internet. Un de ses principes est que le client, par exemple HTTP, est censé vérifier que le nom dans le certificat présenté par le serveur coïncide avec le nom demandé par le client. Et si le serveur a plusieurs noms ?

Ne demandons pas à l'ITU d'avoir prévu un cas aussi simple que celui où une machine est connue par plusieurs noms, comme par exemple www.example.org et www.example.net. Il faut pour cela utiliser des extensions à X.509 telles les "*Subject Alternative Name*" du RFC 5280[1]. Ces extensions sont correctement gérées par certains logiciels tels que OpenSSL mais pas forcément par toutes les autorités de certification.

Voyons maintenant les détails pratiques, très bien expliqués en <http://therowes.net/~greg/2008/01/08/creating-a-certificate-with-multiple-hostnames/> ou en <http://www.hsc.fr/ressources/breves/ssl_virtualhosts.html.fr>.

Comment mettre de tels noms supplémentaires dans une demande de signature (CSR pour "*Certificate Signing Request*") ? Avec OpenSSL, il faut éditer openssl.cnf pour y ajouter ces noms (je ne trouve pas de moyen de les indiquer en ligne de commande, ou de faire qu'OpenSSL les demande interactivement :

```
x509_extensions = v3_req
...
[ v3_req ]
...
subjectAltName          = @alt_names

[alt_names]
# www.example.net sera donné interactivement
DNS.1   = www.example.org
```

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5280.txt

Et il faut vérifier que la CSR est correcte :

```
% openssl req -text -in www.example.org.csr
...
           X509v3 Subject Alternative Name:
                DNS:www.example.org
...
```

La CA qui signe doit elle-même gérer ces extensions. Si elle utilise OpenSSL, elle doit s'assurer d'avoir dans son `openssl.cnf` :

```
copy_extensions = copy
```

Rien de cela n'est spécifique à HTTP. Par exemple, si on protège ses connexions POP avec TLS, un MUA comme Thunderbird vérifie les noms et proteste si sa configuration lui dit de se connecter à `mail.example.org` alors que le certificat X.509 ne contient que `mail.example.net`. L'utilisation de noms supplémentaires résoud le problème, Thunderbird ne râle plus.

Une autre solution à ce problème est possible pour les protocoles qui ne commencent pas TLS tout de suite, qui envoient une requête `STARTTLS` **et** qui transmettent le nom du serveur (pour permettre à celui-ci de déterminer le certificat à utiliser). C'est le cas de HTTP (RFC 2817) comme le rappelle Pierre Beyssac <https://signal.eu.org/blog/2007/09/07/http-et-tls-la-rfc-meconnue/> dans un article qui cite les mises en œuvre possibles (mod_gnutls <http://www.bortzmeyer.org/apache-et-gnutls.html> le gère apparemment).

Pour les protocoles comme POP ou IMAP (RFC 2595), qui ne transmettent pas le nom du serveur, je pense que cette autre solution ne marche pas. Une approche possible serait la "*Server Name Indication*" des extensions TLS normalisées dans le RFC 6066, section 3 (merci à Mathieu Arnold pour l'idée). Cette technique permet de transmettre le nom du serveur dans le premier message TLS et fournit donc une solution générale (là encore, elle est mise en œuvre dans GnuTLS <http://www.outoforder.cc/projects/apache/mod_gnutls/docs/#sni-example>). Voir Adieu RFC 2817, bonjour RFC 3546 <https://signal.eu.org/blog/2008/11/25/adieu-rfc-2817-bonjour-rfc-3546/> pour un exemple de mise en œuvre.