Supervision d'enregistrements DANE

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 28 octobre 2017

http://www.bortzmeyer.org/monitor-dane.html

Le système DANE, normalisé dans le RFC 6698 [1], permet de vérifier l'authenticité du certificat d'un serveur TLS, grâce à des enregistrements signés dans le DNS. Comme ces enregistrements doivent toujours être synchrones avec le certificat, il est prudent de les superviser.

DANE fonctionne en publiant dans le DNS un enregistrement qui résume le certificat utilisé par le serveur TLS. Les certificats changents (renouvellement, révocation) et l'enregistrement dans le DNS doit donc suivre ces changements. En outre, dans la plupart des organisations de taille moyenne ou grosse, ce n'est pas la même équipe qui gère le DNS et les serveurs TLS. Il y a donc un gros potentiel de mécommunication, et c'est pour cela que la supervision est cruciale.

J'utilise le logiciel Icinga pour cette supervision, mais l'essentiel des techniques présentées ici marcherait avec tous les logiciels compatibles avec l'API des Monitoring Plugins <https://www.monitoring-plugins.org/>. Première étape, il nous faut un programme qui teste l'état des enregistrements DANE. Nous allons nous appuyer sur le programme `tlsa`, qui fait partie de hash-slinger <http://people.redhat.com/pwouters/hash-slinger/>. Cet ensemble de programme est dans un paquetage Debian, donc l'installer est aussi simple que :

```
% aptitude install hash-slinger
```

Une fois qu'il est installé, testons-le avec un site Web qui a des enregistrements DANE :

```
% tlsa --verify dns.bortzmeyer.org
SUCCESS (Usage 1 [PKIX-EE]): Certificate offered by the server matches the one mentioned in the TLSA record and
SUCCESS (Usage 1 [PKIX-EE]): Certificate offered by the server matches the one mentioned in the TLSA record and
```

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6698.txt

(Il y a deux lignes car le nom `dns.bortzmeyer.org` a deux adresses IP et, potentiellement, il pourrait s'agir de serveurs HTTP, voire de deux machines différentes.) En prime, `tlsa` définit évidemment le code de retour :

```
% tlsa --verify dns.bortzmeyer.org ; echo $?
...
0
```

Si l'enregistrement TLSA a un problème, le code de retour est différent de zéro :

```
% tlsa --verify dougbarton.us ; echo $?
FAIL: Certificate offered by the server does not match the TLSA record (208.79.90.218)
FAIL: Certificate offered by the server does not match the TLSA record (2607:f2f8:ab14::2)
2
```

Rien de magique dans la commande `tlsa`, on aurait pu faire la vérification manuellement, par exemple pour `mercredifiction.bortzmeyer.org` :

```
% dig TLSA _443._tcp.mercredifiction.bortzmeyer.org
...
;; ANSWER SECTION:
_443._tcp.mercredifiction.bortzmeyer.org. 86400 IN TLSA 1 1 1 (
928DDE55DF4CD94CDCF998C55085FBB5228B561CC237
A122D950260029C5B8C9 )
_443._tcp.mercredifiction.bortzmeyer.org. 86400 IN RRSIG TLSA 8 5 86400 (
20171113165155 20171025071419 50229 bortzmeyer.org.
eRNZfL8ERWhs0SIGYHOIMh1l+tzFFfvDmV7XlPxd8BYa
...
```

Puis comparer avec ce que donne `openssl`. Les commandes exactes à utiliser dépendent évidemment de l'usage et du sélecteur indiqués dans l'enregistrement TLSA, cf. RFC 6698. Par exemple, avec le sélecteur 0 (on prend tout le certificat, pas juste la clé), on pourrait utiliser `openssl x509 -fingerprint`. Ici, avec le sélecteur 1 (seulement la clé) :

```
% openssl s_client -servername mercredifiction.bortzmeyer.org -showcerts -connect mercredifiction.bortzmeyer
         openssl x509  -noout -pubkey | openssl pkey -outform der -pubin  | openssl dgst -sha256
...
(stdin)= 928dde55df4cd94cdcf998c55085fbb5228b561cc237a122d950260029c5b8c9
```

On voit qu'on a le même condensat, `928dde55df4cd94cdcf998c55085fbb5228b561cc237a122d95026002` donc tout va bien. (Ne pas oublier le SNI - RFC 6066, l'option `-servername`, sans laquelle vous risquez de récupérer un mauvais certificat.)

Bien, on a désormais un moyen de tester qu'un serveur TLS a bien un certificat qui correspond à l'enregistrement DANE. Maintenant, il faut emballer tout ça dans un script qui est conforme à l'API des "*monitoring plugins*". Le script (trivial), écrit en shell, est disponible ici (en ligne sur `http://www.bortzmeyer.org/files/check_dane_http.sh`). Testons-le :

`http://www.bortzmeyer.org/monitor-dane.html`

```
% /usr/local/lib/nagios/plugins/check_dane_http -H www.afnic.fr ; echo $?
OK - SUCCESS (Usage 3 [DANE-EE]): Certificate offered by the server matches the TLSA record (192.134.5.24)
SUCCESS (Usage 3 [DANE-EE]): Certificate offered by the server matches the TLSA record (2001:67c:2218:30::24)
0

% /usr/local/lib/nagios/plugins/check_dane_http -H dougbarton.us ; echo $?
DANE server 'dougbarton.us' has a problem: FAIL: Certificate offered by the server does not match the TLSA recor
FAIL: Certificate offered by the server does not match the TLSA record (2607:f2f8:ab14::2)
2

% /usr/local/lib/nagios/plugins/check_dane_http -H ssi.gouv.fr ; echo $?
DANE server 'ssi.gouv.fr' has a problem: Unable to resolve ssi.gouv.fr.: Unsuccessful DNS lookup or no data retu
2
```

Le premier domaine a un enregistrement TLSA et marche bien. Le second, comme vu plus haut, a un enregistrement TLSA mais incorrect. Le troisième n'a pas d'enregistrement TLSA.

On peut désormais configurer Icinga pour utiliser ce script :

```
object CheckCommand "dane_http" {
  command = [ PluginContribDir + "/check_dane_http" ]

  arguments = {
          "-H" = "$dane_domain$"
  }
}

...


apply Service for (dane_domain => config in host.vars.dane_domains) {
  import "generic-service"

  check_command = "dane_http"
  vars.dane_domain = dane_domain
}

...

vars.dane_domains["dns.bortzmeyer.org"] = true
```

(Bon, ça serait mieux si on pouvait spécifier indépendemment le nom à tester, l'adresse IP du serveur, et le nom qui va apparaitre dans l'interface Web d'Icinga. Mais j'étais paresseux et j'ai fait simple.)

Voilà, désormais Icinga surveille périodiquement l'enregistrement DANE et le certificat TLS et tout va bien :

S'il y a un problème, on sera averti et on verra le problème dans l'interface Web d'Icinga. Ici, par exemple, l'enregistrement TLSA était pour le certificat entier (sélecteur 0) et un renouvellement a changé le certificat (mais pas la clé). Le message d'Icinga indiquait :

```
CRITICAL since 00:59
Service: mercredifiction.bortzmeyer.org
Plugin Output
DANE server 'mercredifiction.bortzmeyer.org' has a problem: FAIL: Certificate offered by the server does not mat
FAIL: Certificate offered by the server does not match the TLSA record (2001:4b98:dc0:41:216:3eff:fece:1902)
```

Cet incident a permis de tester que la supervision de DANE marchait bien [Caractère Unicode non montré [2]]. Mais il mérite davantage de discussion. Dans ce cas précis, l'enregistrement TLSA commençait par « 1 0 1 ». Cela veut dire « Usage 1, Sélecteur 0, Correspondance 0 » (voir les explications dans le RFC 6698) ou bien, en utilisant les mnémoniques du RFC 7218 (que `tlsa` affiche), « Usage PKIX-EE, Sélecteur Cert, Correspondance SHA-256 ». Et en français ? L'usage PKIX-EE veut dire que l'enregistrement TLSA doit désigner un certificat qui est à la fois valide selon les règles PKIX (RFC 5280) traditionnelles **et** selon DANE, avec le certificat du serveur, pas celui de l'AC. Ensuite, le sélecteur Cert indique que l'enregistrement TLSA doit être un condensat du certificat entier. Or, le renouvellement d'un certificat change les métadonnées (comme, évidemment, la date d'expiration) et donc le condensat. Dans ces conditions, il vaut mieux utiliser le sélecteur 1 (SPKI), où l'enregistrement TLSA va désormais indiquer uniquement la clé, pas tout le certificat. C'est également l'avis de Let's Encrypt <https://community.letsencrypt.org/t/please-avoid-3-0-1-and-3-0-2-dane-tlsa-records-with-le-certificates/7022>..

Voilà, maintenant que ça marche, prochaine étape : faire la même chose pour SMTP (RFC 7672), `tlsa` sachant gérer le TLS lancé après la connexion. Exemple :

```
% tlsa --verify --port 25 --starttls smtp mail.bortzmeyer.org
SUCCESS (Usage 2 [DANE-TA]): A certificate in the certificate chain offered by the server (including the end
SUCCESS (Usage 2 [DANE-TA]): A certificate in the certificate chain offered by the server (including the end
```

Autres solutions possibles pour tester DANE :

— Notez que tous les tests DANE peuvent être faits avec `openssl` seul, avec ses options `-dane_tlsa_domain` et `-dane_tlsa_rrdata` : ainsi, `openssl s_client -dane_tlsa_domain mercredifiction.bortzmeyer.org -dane_tlsa_rrdata "$(dig +short +nodnssec TLSA _443._tcp.mercredifiction.bortzmeyer.org)" -connect mercredifiction.bortzmey` vous affichera un succès (`DANE TLSA 1 1 1 ...c237a122d950260029c5b8c9 matched EE certificate at depth 0`) et le test sur un domaine à problème (`openssl s_client -dane_tlsa_domain dougbarton.us -dane_tlsa_rrdata "$(dig +short +nodnssec TLSA _443._tcp.dougbarton.us)" -connect dougbarton.us:443`) donnera `Verification error: No matching DANE TLSA records.`

— Dans la bibliothèque getdns <https://getdnsapi.net/>, il y a ce programme <https://github.com/getdnsapi/getdns-python-bindings/blob/master/examples/checkdanecert.py>, que je n'ai pas testé.

— À noter aussi (non testés) checkdanemx <https://github.com/binaryfigments/checkdanemx> (écrit en Go), check_dane <https://github.com/debfx/check_dane> (disponible également sur Icinga Exchange <https://exchange.icinga.com/debfx/check_dane>), danecheck <https://github.com/vdukhovni/danecheck> (écrit en Haskell), et pydane <https://github.com/Starch/pydane> (écrit en Python).

2. Car trop difficile à faire afficher par LaTeX