RFC 8358 : Update to Digital Signatures on Internet-Draft Documents

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 13 mars 2018. Dernière mise à jour le 5 avril 2018

Date de publication du RFC : Mars 2018

https://www.bortzmeyer.org/8358.html

Les "*Internet-Drafts*" sont signés suivant les règles du RFC 5485[1], afin qu'une lectrice ou un lecteur puissent vérifier qu'un "*Internet-Draft*" n'a pas été modifié en cours de route. Ce nouveau RFC modifie légèrement le RFC 5485 sur un seul point : la signature d'"*Internet-Drafts*" qui sont écrits en Unicode.

En effet, depuis le RFC 7997, les RFC ne sont plus forcément en ASCII, ils peuvent intégrer des caractères Unicode. Le premier RFC publié avec ces caractères a été le RFC 8187, en septembre 2017. Bientôt, cela sera également possible pour les "*Internet-Drafts*". Cela affecte forcément les règles de signature, d'où cette légère mise à jour.

Le RFC 5485 normalisait l'utilisation de CMS (RFC 5652) pour le format des signatures. Vous pouvez télécharger ces signatures sur n'importe lequel des sites miroirs <https://www.ietf.org/standards/ids/internet-draft-mirror-sites/>. CMS utilise ASN.1, avec l'obligation d'utiliser l'encodage DER, le seul encodage d'ASN.1 qui ne soit pas ambigu (une seule façon de représenter un texte).

Les "*Internet-Drafts*" sont actuellement tous en texte brut, limité à ASCII. Mais cela ne va pas durer (RFC 7990). Les signatures des "*Internet-Drafts*" sont détachées de l'"*Internet-Draft*" (section 2 de notre RFC), dans un fichier portant le même nom auquel on ajoute l'extension `.p7s` (RFC 5751). Par exemple avec wget, pour récupérer un "*Internet-Draft*" et sa signature :

```
% wget  https://www.ietf.org/id/draft-bortzmeyer-dname-root-05.txt
% wget  https://www.ietf.org/id/draft-bortzmeyer-dname-root-05.txt.p7s
```

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5485.txt

(Ne le faites pas avec un "*Internet-Draft*" trop récent, les signatures n'apparaissent qu'au bout de quelques jours, la clé privée n'est pas en ligne.)

La signature est au format CMS (RFC 5652). Son adaptation aux RFC et "*Internet-Drafts*" est normalisée dans le RFC 5485. Le champ `SignedData.SignerInfo.EncapsulatedContentInfo.eContentType` du CMS identifie le type d'"*Internet-Draft*" signé. Les valeurs possibles figurent dans un registre IANA <https://www.iana.org/assignments/smi-numbers/smi-numbers.xml#security-smime-1>. Il y avait déjà des valeurs comme `id-ct-asciiTextWithCRLF` qui identifiait l'"*Internet-Draft*" classique en texte brut en ASCII, notre RFC ajoute (section 5) `id-ct-utf8TextWithCRLF` (texte brut en UTF-8), `id-ct-htmlWithCRLF` (HTML) et `id-ct-epub` (EPUB). Chacun de ces types a un OID, par exemple le texte brut en UTF-8 sera `1.2.840.113549.1.9.16.1.37`.

Maintenant, passons à un morceau délicat, la canonicalisation des "*Internet-Drafts*". Signer nécessite de canonicaliser, autrement, deux textes identiques aux yeux de la lectrice pourraient avoir des signatures différentes. Pour le texte brut en ASCII, le principal problème est celui des fins de ligne, qui peuvent être représentées différemment selon le système d'exploitation. Nous utilisons donc la canonicalisation traditionnelle des fichiers texte sur l'Internet, celle de FTP : le saut de ligne est représenté par deux caractères, CR et LF. Cette forme est souvent connue sous le nom de NVT ("*Network Virtual Terminal*") mais, bien que très ancienne, n'avait été formellement décrite qu'en 2008, dans l'annexe B du RFC 5198, qui traitait pourtant un autre sujet.

Pour les "*Internet-Drafts*" au format XML, notre RFC renvoie simplement au W3C et à sa norme XML, section 2.11 de la cinquième édition <https://www.w3.org/TR/2008/REC-xml-20081126/>, qui dit qu'il faut "*translating both the two-character sequence #xD #xA and any #xD that is not followed by #xA to a single #xA character*". La canonicalisation XML (telle que faite par `xmllint --c14n`) n'est pas prévue.

Les autres formats ne subissent aucune opération particulière de canonicalisation. Un fichier EPUB, par exemple, est considéré comme une simple suite d'octets. On notera que le texte brut en Unicode ne subit pas de normalisation Unicode. C'est sans doute à cause du fait que le RFC 7997, dans sa section 4, considère que c'est hors-sujet. (Ce qui m'a toujours semblé une drôle d'idée, d'autant plus qu'il existe une norme Internet sur la canonicalisation du texte brut en Unicode, RFC 5198, qui impose la normalisation NFC.)

À l'heure actuelle, les "*Internet-Drafts*" sont signés, les outils doivent encore être adaptés aux nouvelles règles de ce RFC, mais elles sont simples et ça ne devrait pas être trop dur. Pour vérifier les signatures, la procédure (qui est documentée <https://www.ietf.org/standards/ids/idsignatures/>) consiste d'abord à installer le logiciel de canonicalisation :

```
% wget  https://www.ietf.org/id-info/canon.c
% make canon
```

Puis à télécharger les certificats racine :

```
% wget  https://www.ietf.org/id-info/verifybundle.pem
```

Vous pouvez examiner ce groupe de certificats avec :

```
% openssl crl2pkcs7 -nocrl -certfile verifybundle.pem  | openssl pkcs7 -print_certs -text
```

Téléchargez ensuite des "*Internet-Drafts*" par exemple en `https://www.ietf.org/id/` :

```
% wget      https://www.ietf.org/id/draft-ietf-isis-sr-yang-03.txt
% wget         https://www.ietf.org/id/draft-ietf-isis-sr-yang-03.txt.p7s
```

On doit ensuite canonicaliser l'"*Internet-Draft*" :

```
% ./canon draft-ietf-isis-sr-yang-03.txt draft-ietf-isis-sr-yang-03.txt.canon
```

On peut alors vérifier les signatures :

```
% openssl cms -binary -verify -CAfile verifybundle.pem -content draft-ietf-isis-sr-yang-03.txt.canon -inform DER
Verification successful
```

Si vous avez à la place :

```
Verification failure
139818719196416:error:2E09A09E:CMS routines:CMS_SignerInfo_verify_content:verification failure:../crypto/cms/cms
139818719196416:error:2E09D06D:CMS routines:CMS_verify:content verify error:../crypto/cms/cms_smime.c:393:
```

c'est sans doute que vous avez oublié l'option `-binary`.

Si vous trouvez la procédure compliquée, il y a un script qui automatise tout ça `idsigcheck` :

```
% ./idsigcheck --setup
% ./idsigcheck draft-ietf-isis-sr-yang-03.txt
```

Si ça vous fait `bad interpreter: /bin/bash^M`, il faut recoder les sauts de ligne :

```
% dos2unix idsigcheck
dos2unix: converting file idsigcheck to Unix format...
```

Ce script appelle OpenSSL mais pas avec les bonnes options à l'heure actuelle, vous risquez donc d'avoir la même erreur que ci-dessus.