Formats des données de PKI

Contenu

Introduction

Ce document décrit les formats des données et les codages d'Infrastructure à clés publiques (PKI) les plus communs.

Conditions préalables

Conditions requises

Cisco vous recommande de prendre connaissance des rubriques suivantes :

- cryptographie à clé publique (concepts de base).
- infrastructure de clé publique (concepts de base).

Composants utilisés

Ce document n'est pas limité à des versions de matériel et de logiciel spécifiques.

Les informations contenues dans ce document ont été créées à partir des périphériques d'un environnement de laboratoire spécifique. Tous les périphériques utilisés dans ce document ont démarré avec une configuration effacée (par défaut). Si votre réseau est opérationnel, assurez-vous que vous comprenez l'effet potentiel de toute commande.

Conventions

Pour plus d'informations sur les conventions utilisées dans ce document, reportez-vous à Conventions relatives aux conseils techniques Cisco.

Notation ASN.1

L'Abstract Syntax Notation One (ASN.1) est un langage formel pour la définition des types et des valeurs de données, et comment ces types et valeurs de données sont utilisés et combinés dans diverses structures de données. Le but de la norme est de définir l'abstract syntax des informations sans contraindre comment les informations sont encodées pour la transmission.

Voici un exemple extrait du *RFC X.509* :

```
   Version  ::=  INTEGER  {  v1(0), v2(1), v3(2)  }
CertificateSerialNumber ::= INTEGER
Validity ::= SEQUENCE {
notBefore Time,
notAfter Time }
Time ::= CHOICE {
utcTime UTCTime,
generalTime GeneralizedTime }
```

Référez-vous à ces documents des sites de normes de l'Union internationale des télécommunications (ITU-T) :

- X.680 ASN.1 : Spécification de notation de base
- X.681 ASN.1 : Spécification d'objet de l'information
- X.682 ASN.1 : Spécification de contrainte
- X.683 ASN.1 : Paramétrisation des caractéristiques ASN.1

Recherche de recommandations ITU-T - Recherchez **X.509** dans **Rec. ou norme** avec le **langage** réglé à **ASN.1**.

Codages BER/CER/DER

L'ITU-T a défini une méthode standard des structures de données de codage décrites dans ASN.1 dans des données binaires. X.690 définit des règles d'encodage simple (JUJUBES) et ses deux sous-ensembles, règles canoniques de codage (CER) et règles distinguées de codage (DER). Chacun des trois est basé sur des zones d'information de type-longueur-**valeur** emballées dans une structure hiérarchique, qui est établie des **ordres**, des **positionnements**, et des **choix**, avec ces différences :

- Le JUJUBE fournit de plusieurs manières d'encoder les mêmes données, qui pas approprié à de cryptos exécutions.
- Le CER fournit le codage clair et utilise des données indéfinies de longueur, avec un repère de fin de données dans des cas spécifiques.
- DER fournit le codage clair et utilise les balises explicites de longueur dans des cas spécifiques.

- Parmi les trois, DER est celui qui est habituellement produit en traitant le PKI et les cryptos charges utiles.

Exemple : Dans DER, 20-bit la valeur `1010 1011 1100 1101 1110` est encodée en tant que :

- **balise** : `0x03` (bitstring)
- **longueur** : `0x04` (octets)
- **valeur** : `0x04ABCDE 0`
- **codage complet DER** : `0x030404ABCDE0`

Les 04 principaux signifie que les 4 derniers bits (égale les **0** chiffres de remorquage) de la valeur encodée doivent être jetés parce que la valeur encodée ne finit pas sur une borne d'octet.

Référez-vous à ces documents du site de normes TU-T :

- Règles de codage X.690 ASN.1 : La spécification des règles d'encodage simple (JUJUBES), des règles canoniques de codage (CER) et du codage distingué ordonne (DER)

Du site de Wikipedia, référez-vous à ces documents :

- Règles d'encodage simple
- Règles canoniques de codage
- Règles distinguées de codage

Vidage hexadécimal DER

Le Cisco IOS, l'appliance de sécurité adaptable (ASA), et d'autres périphériques affichent le contenu DER comme **vidage hexadécimal** avec la **commande show running-config**. Voici la sortie :

```
crypto pki certificate chain root
certificate ca 01
30820213 3082017C A0030201 02020101 300D0609 2A864886 F70D0101 04050030
1D310C30 0A060355 040B1303 54414331 0D300B06 03550403 1304726F 6F74301E
170D3039 30373235 31313436 33325A17 0D313230 37323431 31343633 325A301D
...
```

Ce genre de vidage hexadécimal peut être converti de nouveau à DER dans diverses manières. Par exemple, vous pouvez enlever les caractères espace et les siffler au **programme de xxd** :

```
$ cat ca.hex | tr -d ' ' | xxd -r -p -c 32 | openssl x509 -inform der -text -noout
```

Une autre méthode facile est d'utiliser ce script Perl :

```
#!/usr/bin/perl
foreach (<>) {
s/[^a-fA-F0-9]//g;
print join("", pack("H*", $_));
}
```

```
$ perl hex2der.pl < hex-file.txt > der-file.der
```

En outre, une manière compacte de convertir des **vidages mémoire de CERT**, chacun a précédemment manuellement copié sur un fichier avec l'extension **.hex**, d'une ligne de commande de **coup** comme affiché ici :

```
for hex in *.hex; do
b="${hex%.hex}"
hex2der.pl < "$hex" > "$b".der
openssl x509 -inform der -in "$b".der > "$b".pem
openssl x509 -in "$b".pem -text -noout > "$b".txt
done
```

Chaque fichier a comme conséquence :

- **file.hex -** Le fichier d'origine (doit contenir des chiffres hexadécimaux seulement).
- **file.der - Certificat** dans le format DER (binaire).
- **file.pem - Certificat** dans le format PEM (Base64 + en-tête bas de page).
- **file.txt -** Version conviviale et accessible en lecture du certificat.

Codage Base64

Le codage Base64 représente les données binaires avec seulement 64 caractères imprimables (`A-Za-z0-9+/`) pareillement à l'**uuencode**. Dans la conversion de la binaire en Base64, chaque bloc 6-bit des données d'origine est encodé dans un caractère ASCII imprimable de 8 bits avec une table de traduction. Par conséquent, la taille des données après encoder a augmenté de 33 pour cent (données chronomètre 8 divisés par 6 bits, égaux 1.333).

Une mémoire tampon 24-bit est utilisée pour la traduction de trois (3) groupes de huit (8) bits dans quatre (4) groupes de six (6) bits. Par conséquent un (1) ou deux (2) octets de remplissage pourraient être exigés à la fin du flux de données en entrée. La remplissage est indiquée à la fin des données Base64-encoded, par les égaux un (`=`) signent pour des bits de chaque remplissage du Groupe des Huit (8) ajoutés à l'entrée pendant le codage.

Référez-vous à cet exemple de Wikipedia.

Référez-vous à la plupart d'informations récentes dans RFC 4648 : Les codages des données Base16, Base32, et Base64.

Codage PEM

Le Privacy Enhanced Mail (PEM) est une pleine norme de PKI de l'Internet Engineering Task Force (IETF) afin de permuter les messages sécurisés. Il n'est plus très utilisé en soi, mais sa syntaxe d'encapsulation a été largement empruntée afin de formater et permuter des données liées à la PKI Base64-encoded.

Le RFC 1421 PEM, section 4.4 : Le mécanisme d'encapsulation, définit des messages PEM comme délimités par des bornes d'encapsulation (EBs), qui sont basés sur RFC 934, avec ce format :

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
Header: value
Header: value
...

Base64-encoded data
...
```

```
-----END PRIVACY-ENHANCED MESSAGE-----
```

Dans la pratique aujourd'hui, quand des données PEM-formatées sont distribuées, ce format de borne est utilisé :

```
-----BEGIN type-----
...
-----END type-----
```

le type peut être avec d'autres clés ou Certificats comme :

- CLÉ PRIVÉE RSA
- CLÉ PRIVÉE CHIFFRÉE
- CERTIFICAT
- DEMANDE DE CERTIFICAT
- X509 CRL

Note: Bien que les RFC ne fassent pas cet obligatoire, le nombre de tirets principaux et de remorquages (-) dans l'EBs est significatif et devrait toujours être cinq (5). Autrement, quelques applications, telles qu'OpenSSL, obstruent sur l'entrée. D'autre part, d'autres applications, telles que le Cisco IOS, n'exigent pas EBs du tout.

Référez-vous à ces RFC les plus récents pour de plus amples informations :

- RFC 1421 : Partie I PEM : Procédures de cryptage et d'authentification de message
- RFC 1422 : Partie II PEM : Gestion des clés basée sur certificat
- RFC 1423 : Partie III PEM : Algorithmes, modes, et identifiants
- RFC 1424 : Partie IV PEM : Certification et services connexes principaux

Certificats X.509 et CRLs

X.509 est un sous-ensemble de X.500, qui est une spécification étendue ITU au sujet d'Open Systems Interconnection. Il traite spécifiquement des Certificats et des clés publiques et a été adapté comme norme Internet par l'IETF. X.509 fournit à une structure et à une syntaxe, exprimées en RFC la notation ASN.1, afin d'enregistrer les informations de certificat et des listes des révocations de certificat.

Dans un PKI X.509, un CA délivre un certificat qui lie une clé publique, par exemple : un Rivest-Shamir-Adleman (RSA) ou clé de l'algorithme de signature numérique (DSA) à un nom unique particulier (DN), ou à un nom alternatif tel qu'une adresse e-mail ou un nom de domaine complet (FQDN). Le DN suit la structure dans les normes X.500. Voici un exemple :

```
CN=common-name, OU=organizational-unit, O=organization, L=location,
C=country
```

En raison de la définition ASN.1, les données X.509 peuvent être encodées dans DER afin de pour être permutées en forme binaire, et sur option, convertie en Base64/PEM pour des moyens de communication basés par texte, tels que la copie-pâte sur un terminal.

- Référez-vous à cet Open Systems Interconnection du document X.509 de normes ITU-T - le répertoire : Cadres de certificat de clé publique et d'attribut.
- Référez-vous à RFC 5280 : Pour en savoir plus de profil du certificat X.509 et du Liste des révocations de certificat (CRL).

Normes PKCS

Les normes de cryptographie à clé publique (PKCS) sont des caractéristiques des laboratoires RSA qui se sont en partie transformés en des standards de l'industrie. Ceux le plus souvent produits, affaire avec ces thèmes ; cependant, pas tous affaire avec des formats des données.

PKCS#1 (RFC 3347**)** - Couvre les aspects d'implémentation du chiffrement basé sur RSA (cryptos primitifs, schémas de cryptage/signature, syntaxe ASN.1).

PKCS#5 (RFC 2898**)** - Couvre la dérivation de clé basée sur mot de passe.

PKCS#7 (RFC 2315**)** et RFC 3852 **S/MIME -** définit une syntaxe de message pour transmettre signé et des données cryptées et des Certificats associés. Employé souvent simplement comme conteneur pour les Certificats X.509.

PKCS#8- définit une syntaxe de message pour transporter le libellé ou les keypairs chiffrés RSA.

PKCS#9 (RFC 2985**)** - Définit les classes d'objets et les attributs d'identité supplémentaires.

PKCS#10 (RFC 2986**)** - Définit une syntaxe de message pour des demandes de signature de certificat (CSRs). Un CSR est envoyé par une entité à un CA et contient les informations à signer par le CA, tel que l'information principale publique, l'identité, et les attributs supplémentaires.

PKCS#12 - Définit un conteneur pour des données relatives de empaquetage de PKI (typiquement, **keypair d'entité + CERT + racine d'entité et les Certificats CA intermédiaires**) dans un fichier unique. C'est une évolution du format de l'échange des données personnelles de Microsoft (PFX).

Référez-vous à ces ressources :

- Article de Wikipedia sur PKCS
- Page de laboratoires RSA sur PKCS

Informations connexes

- Support et documentation techniques - Cisco Systems