Formats de données PKI

Contenu

Introduction

Ce document décrit les formats et les codages de données les plus courants de l'infrastructure à clé publique (ICP).

Conditions préalables

Conditions requises

Cisco vous recommande de prendre connaissance des rubriques suivantes :

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

Components Used

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

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Conventions

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

Notation ASN.1

Syntaxe abstraite La première (ASN.1) est un langage formel pour la définition des types et des valeurs de données, et la façon dont ces types et ces valeurs de données sont utilisés et combinés dans diverses structures de données. L'objectif de la norme est de définir la syntaxe abstraite des informations sans restreindre la façon dont les informations sont codées pour transmission.

Voici un exemple extrait de la *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 }
```

Reportez-vous aux documents suivants provenant des sites de normalisation de l'Union internationale des télécommunications (UIT-T) :

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

Recherche de recommandations ITU-T - Rechercher **X.509** dans **Rec. ou standard** avec **Langue** définie sur **ASN.1**.

Codages BER/CER/DER

L'UIT-T a défini une méthode standard de codage des structures de données décrites dans ASN.1 en données binaires. X.690 définit les règles de codage de base (BER) et ses deux sous-ensembles, les règles de codage canonique (CER) et les règles de codage différencié (DER). Tous trois sont basés sur des champs de données **type-longueur-valeur** regroupés dans une structure hiérarchique, construite à partir de **SEQUENCE**, **SET** et **CHOICE**, avec ces différences :

- BER fournit plusieurs méthodes de codage des mêmes données, ce qui n'est pas adapté aux opérations de chiffrement.
- Le CER fournit un codage sans équivoque et utilise des données de longueur indéfinie, avec un marqueur de fin de données dans des cas spécifiques.
- DER fournit un codage sans équivoque et utilise des balises de longueur explicite dans des cas spécifiques.
- Parmi les trois, DER est celui qui est généralement rencontré lors de l'utilisation de PKI et de

charges utiles de chiffrement.

Exemple : Dans DER, la valeur 20 bits 1010 1011 1100 1101 1110 est codée comme suit :

- **balise** : `0x03` (chaîne de bits)
- **longueur:** `0x04` (octets)
- **valeur:** `0x 04 ABCDE0`
- **codage DER complet** : `0x030404ABCDE0`

Le `04` de début signifie que les 4 derniers bits (égal au `0` chiffre de fin) de la valeur codée doivent être ignorés car la valeur codée ne se termine pas sur une limite d'octets.

Reportez-vous aux documents suivants du site des normes TU-T :

- Règles de codage X.690 ASN.1 : Spécification des règles de codage de base (BER), des règles de codage canonique (CER) et des règles de codage différencié (DER)

Sur le site Wikipedia, reportez-vous à ces documents :

- Règles de codage de base
- Règles de codage canonique
- Règles de codage distinctes

Dump hexadécimal DER

Cisco IOS, Adaptive Security Appliance (ASA) et d'autres périphériques affichent le contenu DER comme un **dump 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 type de vidage hexadécimal peut être reconverti en DER de différentes manières. Par exemple, vous pouvez supprimer les caractères d'espace et les diriger vers le **programme xxd** :

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

Une autre méthode simple consiste à 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 de cert**, chacun d'eux ayant été copié manuellement dans un fichier avec extension **.hex**, à partir d'une ligne de commande **bash** comme indiqué 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 génère :

- **file.hex -** Le fichier d'origine (ne doit contenir que des chiffres hexadécimaux).
- **file.der -** Certificat au format DER (binaire).
- **file.pem -** Certificat au format PEM (Base64 + en-tête/pied de page).
- **file.txt -** Version lisible et conviviale du certificat.

Codage Base64

Le codage Base64 représente les données binaires avec seulement 64 caractères imprimables (`A-Za-z0-9+/`) comme **uuencode**. Lors de la conversion du binaire en Base64, chaque bloc de 6 bits des données d'origine est codé en un caractère ASCII imprimable de 8 bits avec une table de traduction. Par conséquent, la taille des données après le codage a augmenté de 33 % (données multipliées par 8 par 6 bits, soit 1,333).

Une mémoire tampon de 24 bits est utilisée pour la traduction de trois (3) groupes de huit (8) bits en quatre (4) groupes de six (6) bits. Par conséquent, un (1) ou deux (2) octets de remplissage peuvent être requis à la fin du flux de données d'entrée. Le remplissage est indiqué à la fin des données codées en base64, par un signe égal (`=`) pour chaque groupe de huit (8) bits de remplissage ajoutés à l'entrée pendant le codage.

Reportez-vous à cet exemple de Wikipedia.

Reportez-vous aux informations les plus récentes du document RFC 4648 : Codages de données Base16, Base32 et Base64.

Codage PEM

Privacy Enhanced Mail (PEM) est une norme ICP IETF (Internet Engineering Task Force) complète afin d'échanger des messages sécurisés. Il n'est plus largement utilisé en tant que tel, mais sa syntaxe d'encapsulation a été largement empruntée afin de formater et d'échanger des données liées à l'ICP codées en Base64.

Le PEM RFC 1421, section 4.4 : Mécanisme d'encapsulation, définit les messages PEM comme étant délimités par des limites d'encapsulation (EoS), basées sur RFC 934, avec ce format :

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

Base64-encoded data
...
-----END PRIVACY-ENHANCED MESSAGE-----
```

Dans la pratique actuelle, lorsque des données au format PEM sont distribuées, ce format de

frontière est utilisé :

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

type peut être associé à d'autres clés ou certificats, tels que :

- CLÉ PRIVÉE RSA
- CLÉ PRIVÉE CHIFFRÉE
- CERTIFICAT
- requête de certificat
- CRL X509

Note: Bien que les RFC ne rendent pas cela obligatoire, le nombre de tirets de début et de fin (-) dans les EB est important et devrait toujours être de cinq (5). Sinon, certaines applications, telles qu'OpenSSL, étouffent l'entrée. D'autre part, d'autres applications, telles que Cisco IOS, ne nécessitent pas du tout d'EoS.

Pour plus d'informations, reportez-vous aux documents RFC les plus récents :

- RFC 1421 : PEM Partie I : Procédures de chiffrement et d'authentification des messages
- RFC 1422 : Partie II du PEM : Gestion des clés basée sur les certificats
- RFC 1423 : PEM Partie III : Algorithmes, modes et identificateurs
- RFC 1424 : Partie IV du PEM : Principales certifications et services connexes

Certificats X.509 et LCR

X.509 est un sous-ensemble de X.500, qui est une spécification ITU étendue sur l'interconnexion de systèmes ouverts. Il traite spécifiquement des certificats et des clés publiques et a été adapté en tant que norme Internet par l'IETF. X.509 fournit une structure et une syntaxe, exprimées dans le document RFC avec une note ASN.1, afin de stocker les informations de certificat et les listes de révocation de certificat.

Dans une ICP X.509, une autorité de certification émet un certificat qui lie une clé publique, par exemple : une clé RSA (Rivest-Shamir-Adleman) ou DSA (Digital Signature Algorithm) pour un nom distinctif (DN) particulier, ou pour un autre nom tel qu'une adresse e-mail ou un nom de domaine complet (FQDN). Le DN suit la structure des normes X.500. Voici un exemple :

```
CN=common-name, OU=unité-organisation, O=organisation, L=emplacement,
C=pays
```

En raison de la définition ASN.1, les données X.509 peuvent être codées dans DER afin d'être échangées sous forme binaire, et éventuellement converties en Base64/PEM pour des moyens de communication basés sur du texte, comme le copier-coller sur un terminal.

- Reportez-vous à ce document de normes ITU-T X.509 Open Systems Interconnection - The Directory : Cadres de certificat de clé publique et d'attribut.
- Reportez-vous à RFC 5280: Profil de liste de révocation de certificats X.509 pour plus d'informations.

Normes PKCS

Les normes de cryptographie à clé publique (PKCS) sont des spécifications de RSA Labs qui ont en partie évolué en normes industrielles. Ceux qui sont le plus souvent rencontrés traitent de ces sujets ; cependant, tous ne traitent pas des formats de données.

PKCS#1 (RFC 3347**)** - couvre les aspects de mise en oeuvre de la cryptographie basée sur RSA (primitives de chiffrement, schémas de chiffrement/signature, syntaxe ASN.1).

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

PKCS#7 (RFC 2315**)** et **S/MIME** RFC 3852 **-** Définit une syntaxe de message pour transmettre des données signées et cryptées et des certificats associés. Souvent utilisé simplement comme conteneur pour les certificats X.509.

PKCS#8 - Définit une syntaxe de message pour transporter des paires de clés RSA chiffrées ou en texte clair.

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

PKCS#10 (RFC 2986**)** - Définit une syntaxe de message pour les demandes de signature de certificat (CSR). Une CSR est envoyée par une entité à une autorité de certification et contient les informations à signer par l'autorité de certification, telles que les informations de clé publique, l'identité et les attributs supplémentaires.

PKCS#12 - Définit un conteneur pour le conditionnement des données PKI liées (généralement, **paire de clés d'entité + certificat d'entité + certificat d'autorité de certification racine et intermédiaire**) dans un seul fichier. Il s'agit d'une évolution du format d'échange d'informations personnelles (PFX) de Microsoft.

Reportez-vous aux ressources suivantes :

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

Informations connexes

- **Support et documentation techniques - Cisco Systems**