Aperçu simple de Protocol d'inscription de certificat

Contenu

Introduction

Ce document décrit l'inscription de certificat simple Protocol (SCEP), qui est un protocole utilisé pour l'inscription et d'autres exécutions d'Infrastructure à clés publiques (PKI).

Informations générales

SCEP a été initialement développé par Cisco, et est documenté dans une ébauche de l'Internet

Engineering Task Force (IETF).

Ses caractéristiques principales sont :

- Modèle de demande/réponse basé sur le HTTP (OBTENEZ la méthode ; soutien facultatif de méthode de POST)
- Seulement chiffrement basé sur RSA de supports
- Utilisations PKCS#10 comme format de demande de certificat
- Les utilisations PKCS#7 afin de transporter cryptographiquement ont signé/messages cryptés
- Prend en charge l'octroi asynchrone par le serveur, avec l'interrogation régulière par le demandeur
- A limité le support de récupération de Liste des révocations de certificat (CRL) (la méthode préférée est par une requête de point de distribution CRL (CDP), pour des raisons d'évolutivité)
- Ne prend en charge pas la révocation de certificat en ligne (doit être fait off-line par des autres moyens)
- Exige l'utilisation d'un champ de **mot de passe de défi** dans la demande de signature de certificat (CSR), qui doit être partagée seulement entre le serveur et le demandeur

L'inscription et l'utilisation de SCEP suit généralement ce flux des tâches :

1. Obtenez une copie du certificat d'Autorité de certification (CA) et validez-la.
2. Générez un CSR et envoyez-le sécurisé au CA.
3. Votez le serveur SCEP afin de vérifier si le certificat a été signé.
4. Re-inscrivez-vous selon les besoins afin d'obtenir un nouveau certificat avant l'expiration du certificat valable.
5. Récupérez le CRL selon les besoins.

Authentification par certificat

SCEP emploie le certificat de CA afin de sécuriser l'échange de message pour le CSR. En conséquence, il est nécessaire d'obtenir une copie du certificat de CA. L'exécution de **GetCACert** est utilisée.

Demande

La demande est envoyée comme requête HTTP GET. Une capture de paquet pour la demande semble semblable à ceci :

```
GET /cgi-bin/pkiclient.exe?operation=GetCACert
```

Réponse

La réponse est simplement le certificat de CA binaire-encodé (X.509). Le client doit valider que le certificat de CA est de confiance par un examen de l'empreinte digital/des informations parasites. Ceci doit être fait par l'intermédiaire d'une méthode hors bande (un appel téléphonique à un administrateur système ou de la pré-configuration de l'empreinte digital dans le point de confiance).

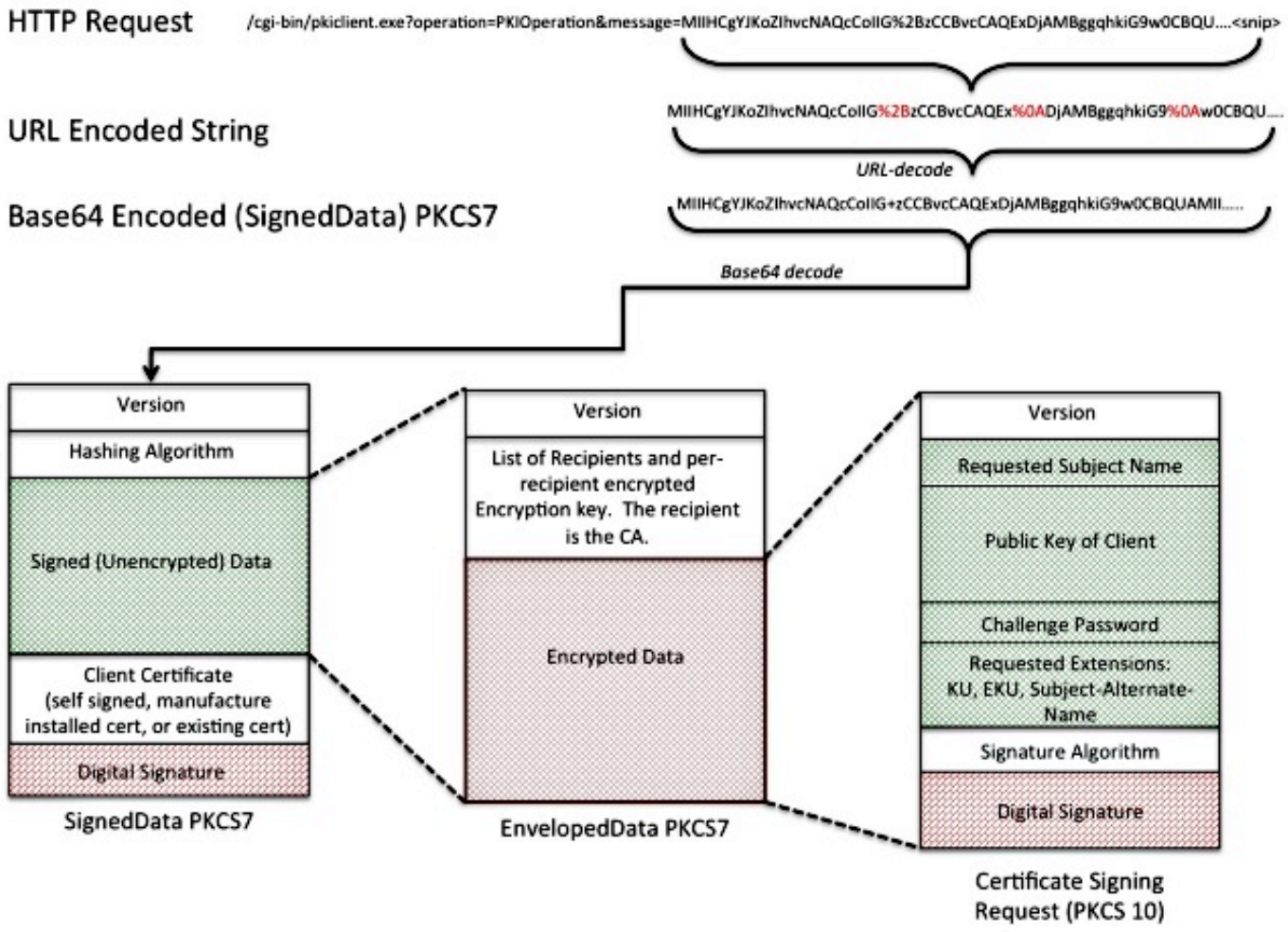
Inscription de client

Demande

La demande d'inscription est envoyée comme requête HTTP GET. Une capture de paquet pour la demande semble semblable à ceci :

```
/cgi-bin/pkiclient.exe?operation=PKIOperation&message=
MIIHCgYJKoZIhvcNAQcCoIIG%2BzCCBvcCAQExDjA......<snip>
```

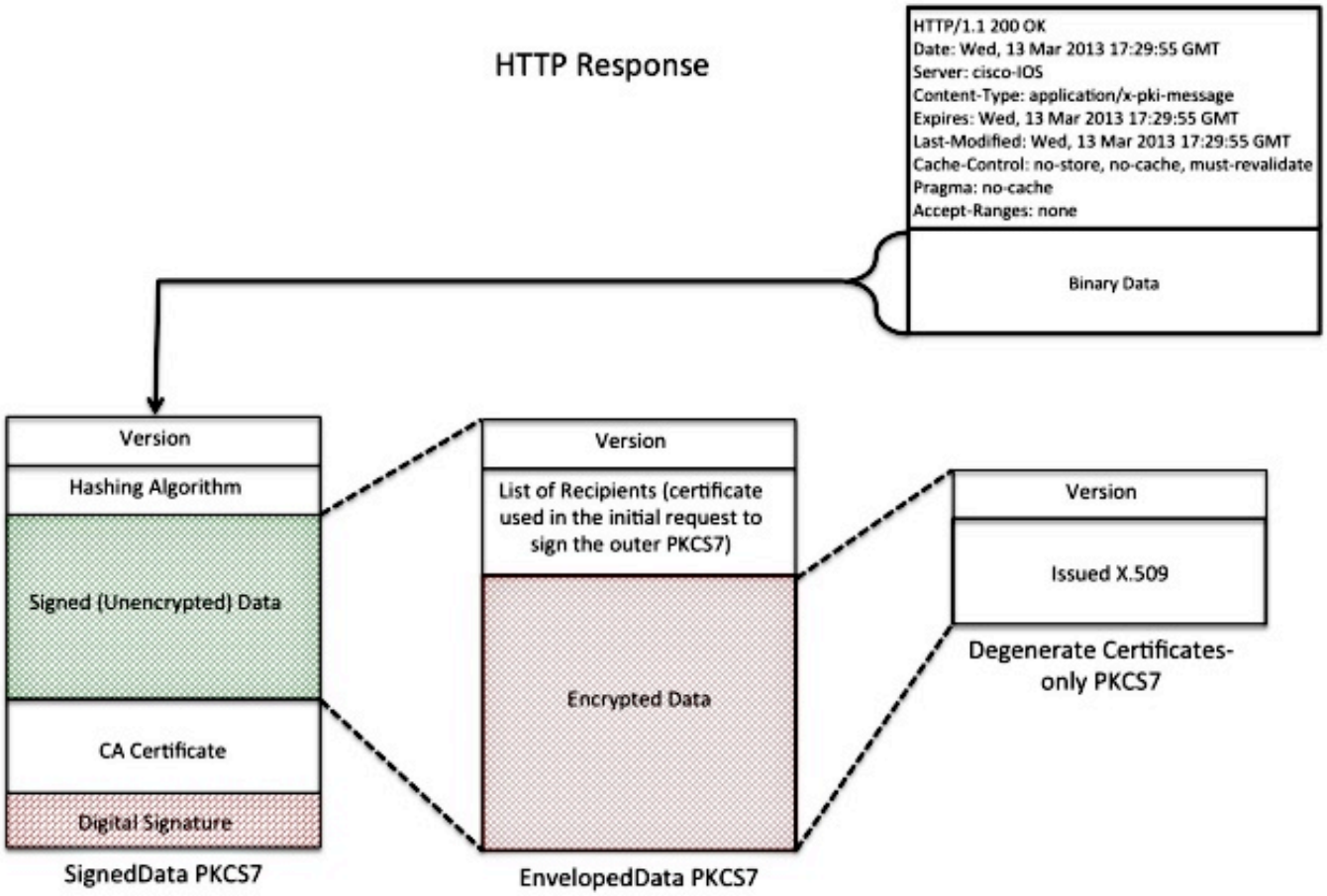
1. Le texte après que le « message= » soit une chaîne encodée par URL, qui est extraite de la chaîne de demande GET.
2. Le texte est alors URL décodé dans une chaîne de texte ASCII. Cette chaîne de texte est un Base64-encoded SignedData PKCS#7.
3. Le SignedData PKCS#7 est signé par le client avec un de ces Certificats ; on l'utilise pour montrer que le client l'a envoyé et qu'il n'a pas été modifié en transit :
   Un certificat auto-signé (utilisé sur l'inscription initiale)Un certificat installé par fabricant (MIC)Une certification en cours qui expire bientôt (la re-inscription)
4. La partie « de données signées » du SignedData PKCS#7 est un EnvelopedData PKCS#7.
5. L'EnvelopedData PKCS#7 est un conteneur qui contient des « données cryptées » et la « clé de déchiffrement. » La clé de déchiffrement est chiffrée avec la clé publique du destinataire. Dans ce cas spécifique, le destinataire est le CA ; en conséquence. Seulement le CA peut réellement déchiffrer les « données cryptées. »
6. La partie de « données cryptées » du PKCS#7 enveloppé est le CSR (PKCS#10).



Réponse

La réponse à la demande d'inscription SCEP est l'un de trois types :

- **Anomalie** - La demande est rejetée par l'administrateur pour un certain nombre de raisons, comme :
  Taille de clé non valideMot de passe de défi non valideLe CA n'a pas pu valider la demandeLa demande a demandé les attributs que le CA n'a pas autorisésLa demande a été signée par une identité à la laquelle le CA ne fait pas confiance
- **En attendant** - L'administrateur CA n'a pas passé en revue la demande encore.
- **Succès** - La demande est reçue et le certificat signé est inclus. Le certificat signé est tenu dans un type particulier de PKCS#7 appelé « un PKCS#7," réservé aux Certificats dégénéré qui est un conteneur spécial qui peut tenir un ou plusieurs X.509 ou CRLs, mais ne contient pas une charge utile signée ou de données cryptées.



Re-inscription de client

Avant l'expiration de certificat, le client doit obtenir un nouveau certificat. Il y a une légère différence comportementale entre le renouvellement et le renversement. Le renouvellement se produit quand le certificat d'ID du client approche l'expiration, et sa date d'expiration n'est pas identique (plus tôt que) que la date d'expiration du certificat de CA. Le renversement se produit quand le certificat d'ID approche l'expiration, et sa date d'expiration est identique que la date d'expiration du certificat de Ca.

Renouvellement

Car la date d'expiration d'un certificat d'ID s'approche, un client SCEP pourrait vouloir obtenir un nouveau certificat. Le client génère un CSR et passe par le procédé d'inscription (comme défini précédemment). Le certificat valable est utilisé afin de signer le SignedData PKCS#7, qui prouve consécutivement l'identité au CA dès réception du nouveau certificat, le client supprime immédiatement le certificat valable et le remplace par le neuf, dont les débuts de validité immédiatement.

Renversement

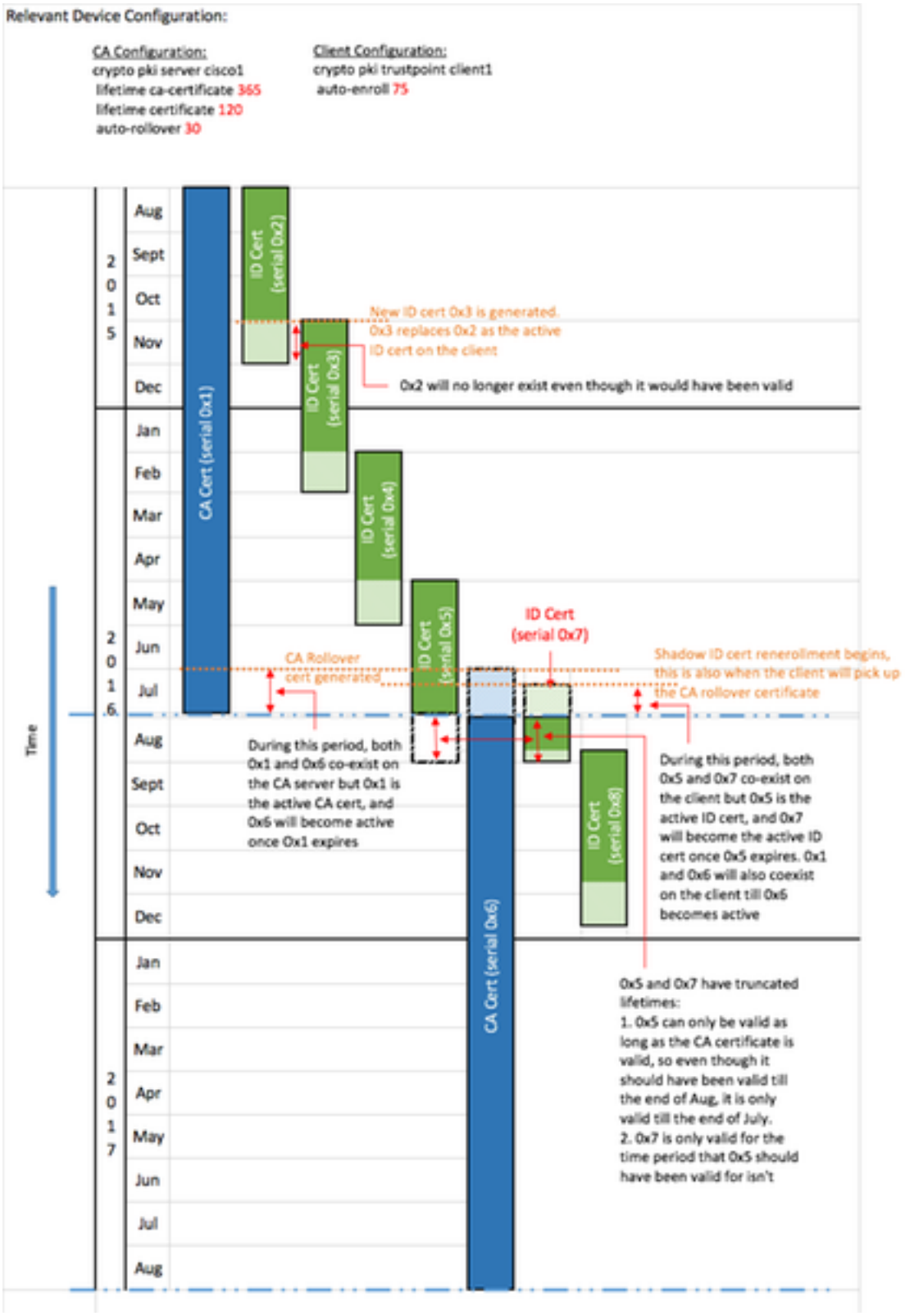
Le renversement est un cas particulier où le certificat de CA expire et un nouveau certificat de CA est généré. Le CA génère un nouveau certificat de CA qui devient valide une fois le certificat de CA en cours expire. Le CA génère habituellement ce certificat du « shadow CA » une certaine

heure avant le temps inversé, parce qu'il est nécessaire afin de générer des Certificats « d'ID de shadow » pour les clients.

Quand le certificat de l'ID du client SCEP approche l'expiration, le client SCEP questionne le CA pour le certificat du « shadow CA ». Ceci est fait avec l'exécution de **GetNextCACert** comme affiché ici :

```
GET /cgi-bin/pkiclient.exe?operation=GetNextCACert
```

Une fois que le client SCEP a le certificat du « shadow CA », il demande un certificat « d'ID de shadow » après la procédure normale d'inscription. Le CA signe le certificat « d'ID de shadow » avec le certificat du « shadow CA ». À la différence d'une demande normale de renouvellement, le certificat « d'ID de shadow » qui est renvoyé devient valide au moment de l'expiration de certificat de CA (renversement). En conséquence, le client doit garder une copie des Certificats pre-- et de POST-renversement pour le CA et le certificat d'ID. Au moment de l'expiration CA (renversement), le client SCEP supprime le certificat de CA et le certificat en cours d'ID et les remplace par les copies de « shadow ».



Modules

Cette structure est utilisée comme modules de SCEP.

Note: PKCS#7 et PKCS#10 ne sont pas SCEP-particularité.

PKCS#7

PKCS#7 est un format des données défini qui permet des données à signer ou être chiffrées. Le format des données inclut les données d'origine et les métadonnées associées nécessaires afin d'exécuter l'exécution cryptographique.

Enveloppe signée (SignedData)

L'enveloppe signée est un format qui porte des données et confirme que les données encapsulées ne sont pas modifiées en transit par l'intermédiaire des signatures numériques. Il inclut ces informations :

```
SignedData &colon;:= SEQUENCE {
version CMSVersion,
digestAlgorithms DigestAlgorithmIdentifiers,
encapContentInfo EncapsulatedContentInfo,
certificates [0] IMPLICIT CertificateSet OPTIONAL,
crls [1] IMPLICIT RevocationInfoChoices OPTIONAL,
signerInfos SignerInfos }
```

- Numéro de version - Avec SCEP, version 1 utilisée.
- Liste d'algorithmes de condensé utilisés - Avec SCEP, il y a seulement un signataire et ainsi seulement un algorithme de hachage.
- Données réelles qui sont signées - Avec SCEP, c'est un format des Envelopper-données PKCS#7 (enveloppe chiffrée).
- Liste de Certificats des signataires - Avec SCEP, c'est un certificat auto-signé sur l'inscription initiale ou le certificat valable si vous re-vous inscrivez.
- Liste des signataires et de l'empreinte digital générés par chaque signataire - avec SCEP, il y a seulement un signataire.

Les données encapsulées ne sont pas chiffrées ou sont assombries. Ce format assure simplement la protection contre le message qui est modifié.

Données enveloppées (EnvelopedData)

Le format des données enveloppé porte les données qui sont chiffrées et peuvent seulement être déchiffrées par les destinataires spécifiés. Il inclut ces informations :

```
EnvelopedData &colon;:= SEQUENCE {
version CMSVersion,
originatorInfo [0] IMPLICIT OriginatorInfo OPTIONAL,
recipientInfos RecipientInfos,
encryptedContentInfo EncryptedContentInfo,
unprotectedAttrs [1] IMPLICIT UnprotectedAttributes OPTIONAL }
```

- Numéro de version - Avec SCEP, la version 0 est utilisée.
- Liste de chacun des destinataires et du Data Encryption Key chiffré relatif - avec SCEP, il y a seulement un destinataire (pour des demandes : le serveur CA ; pour des réponses : le client).
- Les données cryptées - Ceci est chiffré avec aléatoirement une clé générée (qui a été chiffrée avec la clé publique du destinataire).

PKCS#10

PKCS#10 décrit le format d'un CSR. Un CSR contient les informations que les clients demandent

soient inclus dans leurs Certificats :

- Nom du sujet
- Une copie de la clé publique
- Un mot de passe de défi (facultatif)
- Toutes extensions de certificat reqested, comme :
  Utilisation principale (KU)Utilisation principale étendue (EKU)Nom alternatif soumis (SAN)Nom principal universel (UPN)
- Une empreinte digital de la demande

Voici un exemple d'un CSR :

```
Certificate Request:
Data&colon;
Version: 0 (0x0)
Subject: CN=scepclient
Subject Public Key Info:

Public Key Algorithm: rsaEncryption Public-Key: (1024 bit)
Modulus:
00:cd:46:5b:e2:13:f9:bf:14:11:25:6d:ff:2f:43:
64:75:89:77:f6:8a:98:46:97:13:ca:50:83:bb:10:
cf:73:a4:bc:c1:b0:4b:5c:8b:58:25:38:d1:19:00:
a2:35:73:ef:9e:30:72:27:02:b1:64:41:f8:f6:94:
7b:90:c4:04:28:a1:02:c2:20:a2:14:da:b6:42:6f:
e6:cb:bb:33:c4:a3:64:de:4b:3a:7d:4c:a0:d4:e1:
b8:d8:71:cc:c7:59:89:88:43:24:f1:a4:56:66:3f:
10:25:41:69:af:e0:e2:b8:c8:a4:22:89:55:e1:cb:
00:95:31:3f:af:51:3f:53:ad
Exponent: 65537 (0x10001)
Attributes:
challengePassword :
Requested Extensions:
X509v3 Key Usage: critical
Digital Signature, Key Encipherment
X509v3 Subject Alternative Name:
DNS:webserver.example.com
Signature Algorithm: sha1WithRSAEncryption
8c:d6:4c:52:4e:c0:d0:28:ca:cf:dc:c1:67:93:aa:4a:93:d0:
d1:92:d9:66:d0:99:f5:ad:b4:79:a5:da:2d:6a:f0:39:63:8f:
e4:02:b9:bb:39:9d:a0:7a:6e:77:bf:d2:49:22:08:e2:dc:67:
ea:59:45:8f:77:45:60:62:67:64:1d:fe:c7:d6:a0:c3:06:85:
e8:f8:11:54:c5:94:9e:fd:42:69:be:e6:73:40:dc:11:a5:9a:
f5:18:a0:47:33:65:22:d3:45:9f:f0:fd:1d:f4:6f:38:75:c7:
a6:8b:3a:33:07:09:12:f3:f1:af:ba:b7:cf:a6:af:67:cf:47: 60:fc
```

Informations connexes

- Projet soumis à l'IETF SCEP
- Legs SCEP utilisant le guide de configuration CLI
- Configurer le soutien SCEP de BYOD

Annexe

Demandes SCEP

Format de message de demande

Des demandes sont envoyées avec un HTTP OBTIENNENT de la forme :

```
GET CGI-path/pkiclient.exe?operation=operation&message=message HTTP/version
```

Où :

- **le CGI-chemin** dépend du serveur et les points au Common Gateway Interface (CGI) programment que les traitements SCEP demande : Le Cisco IOS® CA utilise une chaîne vide de chemin.Microsoft CA utilise **/certsrv/mscep/mscep.dll**, qui indique le service du service d'inscription de périphérique de réseau MSCEP/(NDES) IIS.
- **L'exécution** identifie l'exécution qui est exécutée.
- **Le message** porte les informations supplémentaires pour cette exécution (et elle peut être vide si aucun réel n'est eu besoin).

Avec la méthode d'OBTENIR, la pièce de **message** est texte brut, ou règles distinguées de codage (DER) - PKCS#7 encodé converti en Base64. Si la méthode de POST est prise en charge, contentez qui serait introduite le codage Base64 avec GET pourrait être introduite le format binaire avec le POST à la place.

Vue schématique

Valeurs possibles pour des **exécutions** et leurs valeurs associées de **message** :

- **exécution =** `PKIOperation` : **messageis une** structure de **pkiMessage** SCEP, basée sur PKCS#7 et encodée avec DER et Base64.la structure de **pkiMessage** peut être de ces types : **PKCSReq** : CSR PKCS#10**GetCertInitial** : vote pour le CSR accordant l'état**GetCert** ou **GetCRL** : certificat ou récupération CRL
- **exécution = GetCACert**, **GetNextCACert**, ou **GetCACaps** (facultatif) : **le message** peut être omis, ou peut être placé à un nom qui identifie le CA.

Réponses SCEP

Format de message de réponse

Des réponses SCEP sont renvoyées en tant que contenu standard de HTTP, avec un **type de contenu** qui dépend de la demande d'origine et du type de données retournés. Le contenu DER est retourné comme binaire (pas dans Base64 quant à la demande). Le contenu PKCS#7 pourrait ou ne pourrait pas contenir données enveloppées chiffrées/signées ; s'il ne fait pas (contient seulement un ensemble de Certificats), il désigné sous le nom d'un PKCS#7 **dégénéré**.

Types satisfaits

Valeurs possibles pour le **type de contenu** :

application/x-pki-message :

- en réponse à l'exécution de **PKIOperation**, avec le **pkiMessage du** type : **PKCSReq,**

GetCertInitial, GetCert ou GetCRL
- le corps de réponse est un **pkiMessage de** type : **CertRep**

application/x-x509-ca-cert :

- en réponse à l'exécution de **GetCACert**
- le corps de réponse est le certificat de CA X.509 DER-encodé

application/x-x509-ca-ra-cert :

- en réponse à l'exécution de **GetCACert**
- le corps de réponse est un PKCS#7 dégénéré DER-encodé qui contient les Certificats CA et de RA

application/x-x509-next-ca-cert :

- en réponse à l'exécution de **GetNextCACert**
- le corps de réponse est une variation d'un **pkiMessage de** type : **CertRep**

La structure de pkiMessage

SCEP OID

```
GET CGI-path/pkiclient.exe?operation=operation&message=message HTTP/version
```

PkiMessage SCEP

- PKCS#7 **SignedData**
- PKCS#7 EnvelopedData (appelé le **pkcsPKIEnvelope** ; facultatif, chiffré dans le destinataire du message)
  messageData (CSR, CERT, CRL,…)
- **SignerInfo** avec des **authenticatedAttributes** :
  transactionID, **messageType**, **senderNoncepkiStatus**, **recipientNonce** (réponse seulement)**failInfo** (réponse + panne seulement)

MessageType SCEP

- demande :
  PKCSReq (19) : CSR PKCS#10**GetCertInitial** (20) : interrogation d'inscription de certificat**GetCert** (21) : récupération de certificat**GetCRL** (22) : Récupération CRL
- réponse :
  CertRep (3) : réponse à délivrer un certificat ou demande CRL

PkiStatus SCEP

- **SUCCÈS** (0) : demande accordée (réponse dans le pkcsPKIEnvelope)
- **PANNE** (2) : demande rejetée (détails dans l'attribut de failInfo)
- **EN ATTENDANT** (3) : la demande attend l'approbation manuelle