Autorité de Certification (CA) et Gestion des Certificats Serveurs

1. Qu'est-ce qu'une Autorité de Certification (CA) ?

Une **Autorité de Certification (CA)** est une organisation ou une entité responsable de délivrer des certificats numériques pour authentifier les identités numériques des serveurs, utilisateurs, ou équipements. Les certificats sont signés avec la clé privée de la CA, garantissant leur authenticité.

---

2. Fonctionnement de la Certification

2.1 Les Certificats X.509

Les certificats délivrés par une CA suivent le standard **X.509** et incluent :

- **Identité** : Informations sur le propriétaire (ex. nom de domaine, organisation).
- **Clé Publique** : Associée à une clé privée détenue par le propriétaire.
- **Autorité de Certification** : Signataire du certificat.
- **Période de Validité** : Dates de début et de fin du certificat.

2.2 Modèle de Confiance

1. Un client (navigateur, logiciel) fait confiance aux **CA racines** préinstallées dans son système.
2. Les CA racines peuvent déléguer à des **CA intermédiaires** pour délivrer les certificats.

2.3 Types de Certificats

1. **Certificats DV (Domain Validation)** : Vérifient uniquement le contrôle du domaine.
2. **Certificats OV (Organization Validation)** : Vérifient l'identité de l'organisation.
3. **Certificats EV (Extended Validation)** : Fournissent un niveau d'authentification élevé (barre verte dans les navigateurs).

---

3. Création et Gestion d'une CA Locale

Pour des besoins internes ou de test, vous pouvez créer votre propre CA et gérer vos certificats.

3.1 Créer une CA Locale

1. **Générer une Clé Privée pour la CA**

```
openssl genrsa -out ca.key 4096
```

2. **Créer un Certificat Racine**

```
openssl req -x509 -new -nodes -key ca.key -sha256 -days 3650 -out
ca.crt
```

- `-x509` : Certificat auto-signé.
- `-days 3650` : Durée de validité (10 ans).
- `ca.crt` : Certificat racine de votre CA.

3.2 Délivrer des Certificats pour les Serveurs

1. **Générer une Clé Privée pour le Serveur**

```
openssl genrsa -out server.key 2048
```

2. **Créer une CSR (Certificate Signing Request)**

```
openssl req -new -key server.key -out server.csr
```

Remplissez les champs importants, comme **Common Name** (nom de domaine).

3. **Signer le Certificat avec votre CA**

```
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -
CAcreateserial -out server.crt -days 365 -sha256
```

- `-CAcreateserial` : Génère un fichier pour suivre les numéros de série des certificats.

4. **Résultat**

- `server.key` : Clé privée du serveur.
- `server.crt` : Certificat signé par la CA.
- `ca.crt` : Certificat racine pour valider le certificat serveur.

3.3 Configurer Apache avec le Certificat

Modifiez votre configuration Apache pour utiliser les certificats :

```
<VirtualHost *:443>
    ServerName example.com

    SSLEngine On
```

```
    SSLCertificateFile /path/to/server.crt
    SSLCertificateKeyFile /path/to/server.key
    SSLCertificateChainFile /path/to/ca.crt

    DocumentRoot /var/www/html

    ErrorLog ${APACHE_LOG_DIR}/ssl_error.log
    CustomLog ${APACHE_LOG_DIR}/ssl_access.log combined
</VirtualHost>
```

Redémarrez Apache :

```
sudo systemctl restart apache2
```

---

4. Gestion des Certificats

4.1 Renouvellement des Certificats

- Les certificats ont une durée limitée (ex. 90 jours pour Let's Encrypt).
- Renouvelez avant l'expiration pour éviter des erreurs de certificat.

4.2 Révocation des Certificats

- Si un certificat est compromis, vous devez le révoquer.
- Utilisez une **CRL (Certificate Revocation List)** ou l'OCSP (Online Certificate Status Protocol).

Créer une Liste de Révocation (CRL) :

1. Révoquez un certificat :

   ```
   openssl ca -revoke /path/to/cert.pem
   ```

2. Générer la CRL :

   ```
   openssl ca -gencrl -out ca.crl
   ```

3. Distribuez la CRL aux systèmes clients.

---

5. Automatisation avec Let's Encrypt

Avantages

1. Certificats gratuits et automatisés.
2. Compatible avec des outils comme Certbot pour le renouvellement automatique.

Configurer Let's Encrypt

1. Installez Certbot :

```
sudo apt install certbot python3-certbot-apache
```

2. Obtenez et configurez un certificat :

```
sudo certbot --apache -d example.com -d www.example.com
```

3. Configurez le renouvellement automatique :

```
sudo certbot renew --quiet
```

---

6. Bonnes Pratiques

1. **Certificats Valides** :
   - Utilisez des certificats signés par une autorité reconnue pour les sites publics.
   - Réservez les CA locales pour les environnements internes.
2. **Surveillance des Certificats** :
   - Utilisez des outils pour surveiller les dates d'expiration (ex. Nagios, Prometheus).
3. **Renforcement de la Sécurité** :
   - Désactivez les protocoles SSLv2, SSLv3, et TLS obsolètes.
   - Utilisez des clés de 2048 bits minimum.
4. **Chaînes de Certificat Complètes** :
   - Fournissez toujours les certificats intermédiaires avec le certificat serveur pour éviter les erreurs de confiance.

---

7. Exemple Complet : CA Locale avec Apache

1. **Créer la CA et le Certificat Serveur**

```
```

```
# Clé privée de la CA
openssl genrsa -out ca.key 4096
# Certificat de la CA
openssl req -x509 -new -nodes -key ca.key -sha256 -days 3650 -out
ca.crt

# Clé privée du serveur
openssl genrsa -out server.key 2048
# CSR pour le serveur
openssl req -new -key server.key -out server.csr
# Signer le certificat
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -
CAcreateserial -out server.crt -days 365 -sha256
```

2. **Configurer Apache**

```
<VirtualHost *:443>
    ServerName example.com

    SSLEngine On
    SSLCertificateFile /etc/ssl/certs/server.crt
    SSLCertificateKeyFile /etc/ssl/private/server.key
    SSLCertificateChainFile /etc/ssl/certs/ca.crt

    DocumentRoot /var/www/html
</VirtualHost>
```

3. **Vérifiez la Configuration**
   Redémarrez Apache et testez le certificat avec un navigateur ou `curl`.

---

8. Résumé

- Une **CA locale** permet de signer vos propres certificats pour un usage interne.
- Pour un usage public, privilégiez des certificats signés par des autorités reconnues comme Let's Encrypt.
- La gestion des certificats implique leur création, renouvellement, et révocation en cas de compromission.
- Apache HTTPD offre une configuration flexible pour intégrer facilement les certificats SSL/TLS.