TP-2 Certificats numériques
A réaliser sous Linux

1.Introduction
Pour initier une transaction sécurisée par SSL, un serveur doit fournir un certificat et une clé publique. Ce certificat doit être signé avec la clé privée du CA afin de garantir son authenticité.

Dans ce TP nous allons supposer que nous avons à organiser des transactions sécurisées dans un intranet d'entreprise réparti sur plusieurs continents. Pour renforcer la sécurité, les serveurs de cet intranet utilisent une connexion SSL. Étant donné que ces serveurs ne sont pas accessibles de l'extérieur on souhaite utiliser une autorité de certification privée au niveau de l'intranet.

Nous allons donc commencer par créer notre CA, puis nous verrons comment les serveurs et les clients peuvent faire des demandes de certificats. Enfin nous verrons comment vérifier, révoquer et utiliser concrètement ces certificats. Toutes ces opérations utiliseront les fonctionnalités d'OpenSSL, qu'elles émanent de la CA, des serveurs ou des clients.

2.Préparation de l'environnement de la CA
L'environnement de la CA est composé de répertoires et de fichiers dédiés. Il comprend aussi un fichier de configuration qui est très proche du fichier de configuration par défaut d'OpenSSL. Pour personnaliser notre CA, nous allons recopier ce fichier standard dans l'environnement de la CA et nous le modifierons pour qu'il corresponde à nos besoins.

1) Créer un répertoire CA qui contiendra les 2 répertoires suivants :
private : les clés privées (pour le CA et pour les certificats usagers).
newcerts : contient les nouveaux certificats créés avec pour nom de fichier un numéro de série

2) Dans CA on créera deux fichiers
i**ndex.txt** : Utilisé par la CA pour stocker les informations sur les certificats signés
serial : chaque certificat possède un numéro de série (serial). Ce fichier contient numéro du prochain certificat incrémentée automatiquement. Pour générer le premier numéro de série, on pourra utiliser la commande suivante : **echo "01" > serial**

3) Adapter le fichier de configuration de la CA
En vous aidant de Google cherchez sur votre machine l'emplacement du fichier de configuration d'OpenSSL (openssl.cnf) puis copier le vers notre répertoire CA en le renommant ca.cnf Ce fichier contient un certains nombres d'information qu'il faut préciser et d'autres qui sont optionnelles. Commencer par analyser le contenu de ce fichier et identifiez la durée de validité par défaut des certificats. Ensuite, modifiez le contenu pour le faire pointer vers le répertoire CA et l'adapter à l'identité de notre CA.

[CA_default]
dir = .

[req_distinguished_name]
countryName_default = FR
stateOrProvinceName_default = NORD
localityName_default = VILLENEUVE ASCQ
0.organizationName_default = SECURITE

Dans le cadre du TP, on se contentera du minimum d'information de personnalisation. Dans la pratique il faudrait être plus complet. Notez que dans les opérations qui suivent, certaines informations vous seront demandées par OpenSSL. Les valeurs par défaut que nous venons de saisir pourront être validées en faisant simplement entrée au clavier. D'autres données pourront être omises (faire simplement entrée). D'autres informations comme les « passphrases » ou le « commonName » devront être renseignées faute de mise en erreur. Ces éléments seront précisés, plus loin au fur et à mesure.

3.Création de la CA

Cette étape consiste à créer le certificat et la clé privée qui servira à signer les certificats émis par la CA. Pour cette opération, on exécutera la commande qui suit dans le répertoire CA

openssl req -new -x509 -extensions v3_ca -newkey rsa:4096 -keyout private/cakey.pem -out cacert.pem -config ca.cnf

req : option de génération de certificats PKCS#10.
-x509 : Génère le certificat auto-signé racine d'une CA
-extensions v3_ca : certificat SSL v3 (section v3_ca du fichier de configuration ca.vnf).
-newkey rsa:4096 : clé privée RSA de 4096 bits.
-keyout private/cakey.pem : fichier contenant la clé privée
-out cacert.pem : fichier avec le certificat
-config ca-config : permet d'utiliser le fichier de configuration ca.cnf. Sans cette option utilisation du fichier par défaut openssl.cnf.

Remarques : (1)L'option **-config** n'est nécessaire que si, comme dans notre cas, on ne souhaite pas utiliser et modifier le fichier de configuration par défaut d'OpenSSL (2)Alternativement, il est aussi possible de générer la clé et le certificat avec deux commandes séparées.

Observez les fichiers créés dans votre environnement CA et leurs contenus. Après cette étape on a donc, les deux éléments de base de l'autorité de certification : lesquels ?

4.Création du CSR (Certificat Signing Request)

Le CSR est émis par un serveur ou un client pour faire une demande de certificat à la CA. Selon que le certificat est utilisé par un serveur ou un client, la démarche est différente. Elle requiert un mot de passe dans le cas de la demande client. Elle n'en nécessite pas dans le cas de la demande serveur (demande faite par une machine et pas un humain), dans ce cas on utilise l'option **-nodes** (clée privée non chiffrée).

Le champ ***commonName***, doit être renseigné. Il doit correspond, par exemple, à l'URL du serveur qui va être sollicité par le client.

On remarquera que la commande serveur demande un challenge password. Ce dernier n'est pas utilisé pour générer la clé privée (comme c'est le cas pour le client à qui on demande un passphrase). Le challenge password est un secret partagé entre votre entité et votre émetteur SSL, encapsulé dans le CSR et qui permet de vous authentifier si c'est nécessaire. Le challenge peut donc être omis contrairement au passphrase qui provoque une erreur s'il est omis.

CSR pour le serveur
openssl req -new -nodes -newkey rsa:2048 -keyout private/webserver.key -out webserver.csr -config ca.cnf

CSR pour le client
openssl req -new -newkey rsa:2048 -keyout private/makeyclient.key -out makeyclient.csr -config ca.cnf

Analysez ces deux requêtes, les fichiers générés et leurs contenus pour bien comprendre le fonctionnement. Listez le type de fichiers générés (clés privés, publiques, certificats ? A qui (issuer ou subject) appartiennent ces fichiers.

5.Création de certificats signés

Les CSR vont permettre de générer les certificats. Nous nous limiterons ici à la création du certificat serveur. Avant de saisir la commande qui permet cette opération faites le tour du contenu de votre CA (fichiers, répertoires, etc). Ceci vous permettra d'évaluer l'incidence de cette commande.

openssl ca -config ca.cnf -policy policy_anything -out webserver.crt -infiles webserver.csr

ca : fonction CA de base permettant de signer des demandes de certificats.
-policy policy_anything : politique de certification de la CA. Voir zones du ca.cnf précisant les champs qui sont obligatoires ou qui doivent correspondre au certificat de la CA.
-out webserver.crt : spécifie le nom du fichier (le certificat) de sortie.
-infiles webserver.csr : fichiers CSR à utiliser.

Éditer et analysez le contenu du fichier index.txt. Quels sont les liens entre ce contenu et les autres fichiers générés à l'occasion de la création de ce certificat ?

Étudiez le certificat du serveur. Quelle est la date limite de validé du certificat ? Qui est l'issuer de ce certificat ? Quelle version d'X509 est utilisée ? Quel algorithme a été utilisé pour calculer le digest ?

6.Visualiser et vérifier les certificats

OpenSSL permet de visualiser différentes informations liées aux certificats. Ceci peut être fait selon le cas par un serveur ou par la CA.

Affichage du contenu d'un certficat
openssl x509 -in webserver.crt -noout -text

x509 : affichage et signature de certificat.
-in webserver.crt : le certificat à lire.
-noout : permet de ne pas afficher le certificat encodé.
-text : afficher toutes les informations du certificat.

Vérification d'un certificat d'une CA
openssl verify -CAfile cacert.pem webserver.crt

verify : utilitaire de vérification de certificats.
-CAfile cacert.pem: le certificat du CA.
webserver.crt : le certificat à vérifier.

Remarques : Pour un système Debian, il suffit de copier le certificat dans le répertoire ***/usr/share/ca-certificates/*** et de reconfigurer le paquet **ca-certificates** (dpkg-reconfigure ca-certificates) afin d'ajouter le certificat à la liste des tiers de confiance. Le fichier sera automatiquement copié dans **/etc/ssl/certs**.

7.Révoquer des certificats

Pour différentes raisons (plaintes, soupçons de piratage, ..), une CA peut souhaiter révoquer des certificats. Cette étape intervient notamment par la gestion d'une CRL (Certficate Revocation List) qui contient les numéros de série des certificats révoqués. Ce fichier va pouvoir être utilisé, par exemple par le serveur

Apache pour refuser des connexions SSL. Il suffira d'ajouter la directive SSLCARevaocationPath pointant vers le fichier CRL dans le fichier de configuration **httpd.conf** d'Apache.

Le système de révocation utilise aussi le fichier **crlnumber** qu'OpenSSL va utiliser pour garder en mémoire le prochain numéro de révocation à utiliser.

Créons le fichier crlnumber
echo 10 > crlnumber

Créons la CRL
openssl ca -keyfile private/cakey.pem -cert cacert.pem -gencrl -out crl.pem -config ca.cnf

Visualiser le contenu de la CRL et notez sa constitution.
openssl crl -in crl.pem -text

Révoquer le certificat de webserver créé à l'étape 5
openssl ca -keyfile private/cakey.pem -cert cacert.pem -revoke webserver.crt -config ca.cnf

Que constatez vous en analysant le fichier index.txt ?

Régénération de la CRL
openssl ca -keyfile private/cakey.pem -cert cacert.pem -gencrl -out crl.pem -config ca.cnf

Visualisez à nouveau le contenu de la CRL, que constatez vous ?

Pour vous familiariser avec le processus de création et de révocation, tester la génération et la révocation de 3 ou 4 certificats puis identifiez l'incidence de ces opérations sur le contenu du répertoire CA et de ses sous répertoires