TP MI44

Objectif

Nous supposons un club privé qui distribue un mot de passe d'entrée aux adhérents une fois par mois. Ils utilisent ce mot de passe pour accéder au club. Les adhérents obtiennent le mot de passe en se connectant sur un lien URL secret. Le lien leur est communiqué lors de leur dernière rencontre physique. Actuellement, il s'agit simplement d'une connexion http, ce qui permet à toute personne observant le trafic de lire le mot de passe du club. L'objectif du projet est de remplacer la connexion http par une connexion https.

Les étapes en résumé

Pour ce faire, le projet consiste à :

1- Créer une autorité de certification (ca) détenant une paire de clés (publique, privée) et un certificat (le tiers de confiance)
2- Générer un certificat du serveur signé par la ca. Ceci se déroule en deux étapes :
    a. Générer le certificat de la requête de certification (CSR : Certificate Signing Request)
    b. Faire signer le certificat par la ca.
3- Remplacer la connexion http par la connexion https

Le rendu

Plusieurs étapes sont nécessaires pour réaliser le projet. Le projet consiste à accomplir chacune de ces étapes. Chaque étape nécessite une réponse sous la forme d'un code ou d'une capture d'écran accompagnée d'une remarque. L'ensemble vous permettra de constituer un rapport accompagné de fichiers de code en python. Le rapport à rendre est sous le format pdf. Le nom du fichier est composé des noms des auteurs (vous pouvez réaliser le projet en binôme). Le projet est à rendre avant le 21 juin sur Moodle.

Fichiers

Ce projet est fourni avec un ensemble de fichiers *.py . Certains fichiers doivent être complétés pour fonctionner. Ceci est indiqué dans l'étape correspondante. Le tableau ci-dessous donne le descriptif de chaque fichier *.py

Tableau 1. Fichiers fournis avec le projet

Fichier	Contenu
serveur.py	Ce fichier permet de réaliser un serveur http pour lire le mot de passe du club. Il doit évoluer pour permettre une connexion https
PKI_utile.py	C'est un code constitué de quatre fonctions principales 1- generate_private_key : génère les clés privées 2- generate_public_key : génère le certificat autosigné de l'autorité de certification (ca) 3- generate_csr : génère le certificat csr de la requête de certification 4- sign_csr : signe le certificat csr par la clé privée du ca.
ca_certificat.py	Ce fichier est à compléter pour permettre de générer la paire de clés de l'autorité de certification. Il en résultera les deux fichiers suivants :

	1- ca-cle-privee.pem : il contient la clé privée de la ca. Cette clé est lisible seulement avec un mot de passe choisi par vos soins. 2- ca-cle-publique.pem : il s'agit du certificat autosigné de la ca. Dans se fichier, on trouve la clé publique de la ca en plus d'autres renseignements.
csr_certificat.py	Ce fichier est à compléter pour permettre de générer la paire de clés du serveur à signer par la ca. Il en résulte les fichiers : 1- serveur-cle-privee.pem : il contient la clé privée du serveur https du club. Cette clé est lisible seulement avec un mot de passe 2- serveur-csr.pem : il s'agit d'une première version du certificat du serveur qui nécessite d'être signé par la ca. Il contient donc la clé publique du serveur certifcat à faire signer la ca.
certificatserveur.py	Ce fichier crée le certificat du serveur 'serveur-cle-publique.pem' signé par la ca.

La figure ci-dessous décrit les étapes avec les fichiers pythons correspondants. Les flèches en jaune montrent les fonctions auxquelles les fichiers pythons font appel. Ces fonctions sont déjà présentes dans PKI_utile.py.

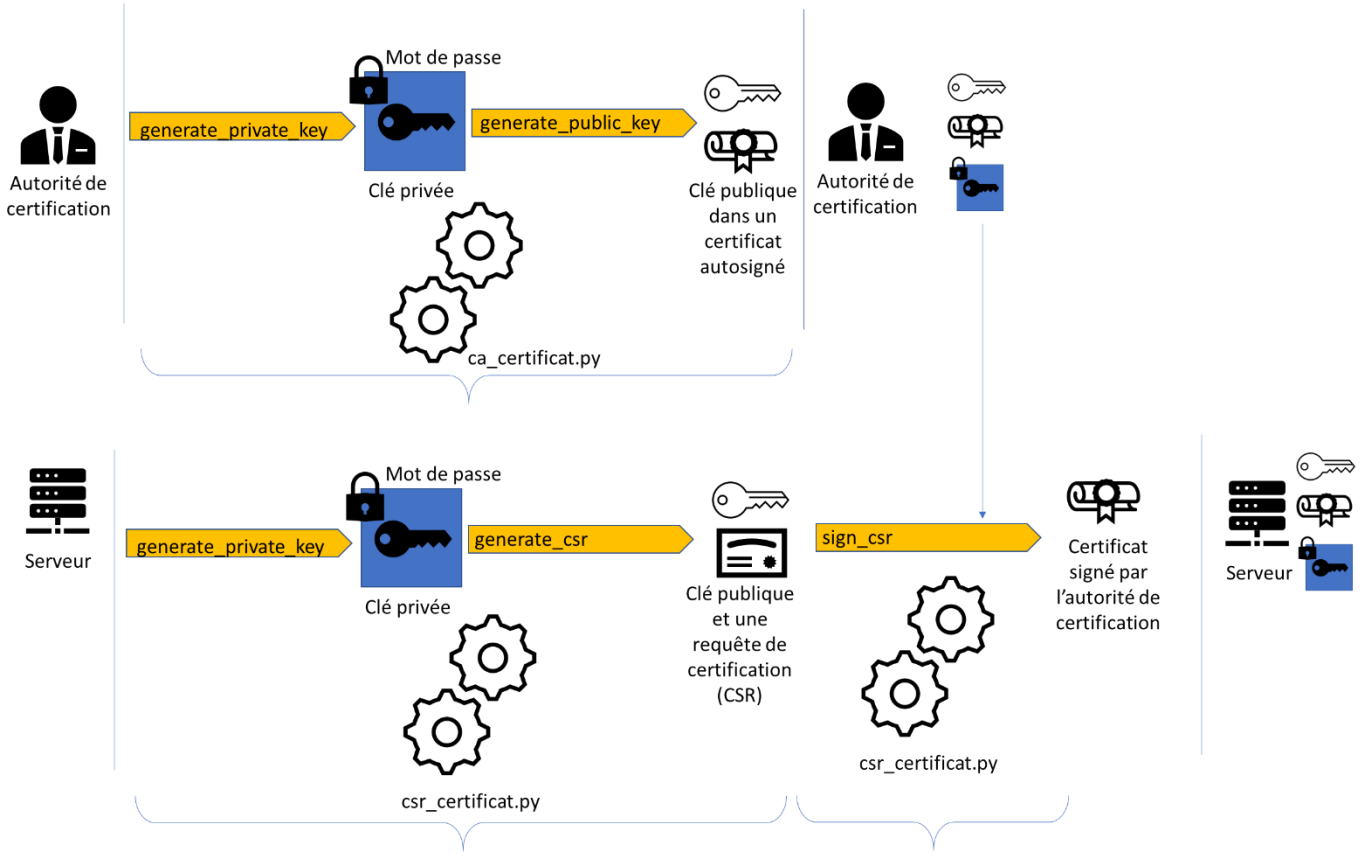

Figure 1. Etapes du projet pour générer le certificat du serveur

Etape 1 : Vérification du serveur http

Voici comment exécuter le serveur http avec anaconda :

1- Flask : Sur le terminal d'anaconda lancer la commande : pip install flask
2- Exécuter la ligne de commande suivante pour lancer le serveur : python seveur.py
3- Aller sur votre navigateur et tapez «localhost:8081».
4- N'oublier pas d'arrêter le test en cliquant simultanément sur les touches Ctrl et c sur le terminal.

Pour se convaincre que tout le monde peut observer le mot de passe, utiliser wireshark. Il est possible de filtrer les messages capturer selon l'interface, le protocole et le port. Pour «écouter le hôte local, il suffit d'écouter le Loopback. Pour pouvoir filtrer selon le port vous tapez tcp.port==8081. Vous cherchez par la suite la ligne contenant dans la partie information « text/html ».

Pour cette étape, vous devez changer le mot de passe fourni et présenter une copie d'écran commentée du navigateur et de wireshark.

Etape 2 : Génération du certificat de l'autorité de certification

Il faut compléter le fichier ca_certificat.py de telle sorte à ce qu'il génère le certificat autosigné de la ca (voir Figure 2). Ainsi vous avez besoin de générer le fichier contenant la clé privée de la ca : ca-cle-privee.pem. Ce fichier n'est accessible qu'à l'aide d'un mot de passe que vous avez défini (introduit en paramètre de la fonction generate_privte_key). Par la suite vous complétez le fichier fichier ca_certificat.py pour pouvoir générer le certificat autosigné nommé ca-cle-publique.pem. Il faut renseigner les paramètres de la fonction public_key avec les données liées à l'autorité de certification (country, state…). Afin de visualiser le certificat autosigné, vous créez un fichier imprime_pem.py qui affiche le contenu des fichiers x-cle-publique.pem. Pour l'affichage vous pouvez utiliser la librairie pem de python (réalisable en 5 lignes de code).

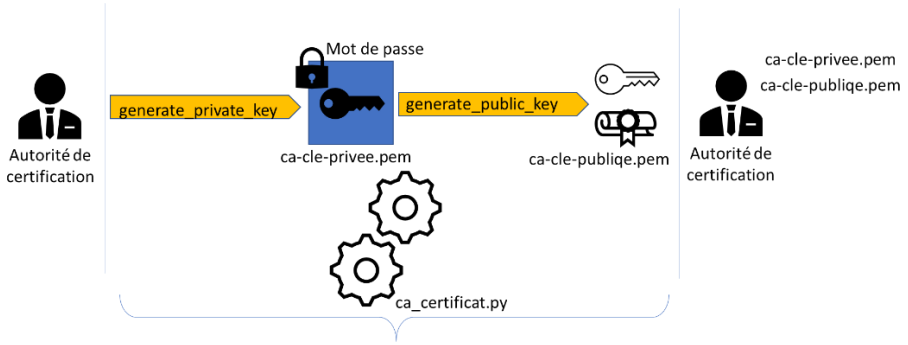

Figure 2. Génération du certificat de l'autorité de certification

Dans cette étape, vous rendez la nouvelle version de ca_certificat.py, le fichier imprime_pem.py, ca-cle-privee.pem et ca-cle-publique.pem. Dans le rapport, vous mettez le résultat de la visualisation du ca-cle-publique.pem, vous indiquez le mot de passe de la clé privée de la ca ainsi que les données de la ca. Vous êtes libre dans le choix de ces données.

Etape 3 : Génération du certificat du serveur.

Dans cette étape vous complétez csr_certificat.py de telle sorte à ce qu'il génère la clé privée du serveur (cle-privee-serveur.pem) ainsi que le certificat csr (serveur-csr.pem) qui contient la clé publique du serveur et vous l'exécuter. Vous exécutez ensuite le fichier certificatserveur.py pour obtenir le certificat définitif du serveur (serveur-cle-publique.pem). Voir les étape en détail dans la Figure 3.

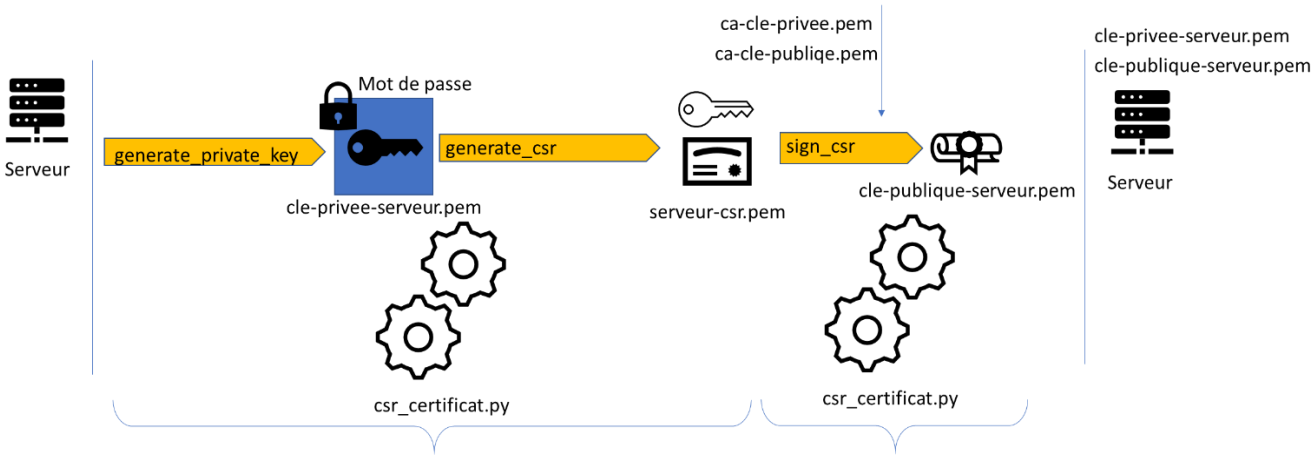

Figure 3. Création du certificat du serveur

Dans cette étape vous rendez la nouvelle version du csr_certificat.py et les fichiers *.pem relatifs au serveur. Le rapport doit contenir le mot de passe permettant la lecture du ficher contenant la clé privée du serveur. Il doit aussi contenir le résultat de la visualisation du certificat du serveur.

Etape 4 : Connexion https

Modifier le fichier serveur.py afin de permettre aux membres du club de se connecter en https. Tester le serveur https sur votre navigateur et lire le message afficher.

Pour cette partie vous devez rendre la nouvelle version serveur.py et commenter le message d'erreur en fournissant une copie d'écran avec la lecture du mot de passe du club. Discutez aussi les solutions à adopter pour éviter le message d'erreur du navigateur.

Etape 5 :

Ajouter quelques améliorations qui vous semblent intéressantes au PKI_utile.py. Soumettre votre version de PKI_utile.py et présenter les améliorations en les justifiants dans le rapport.