CRY 2020

Laboratoire #4

12-01-2021

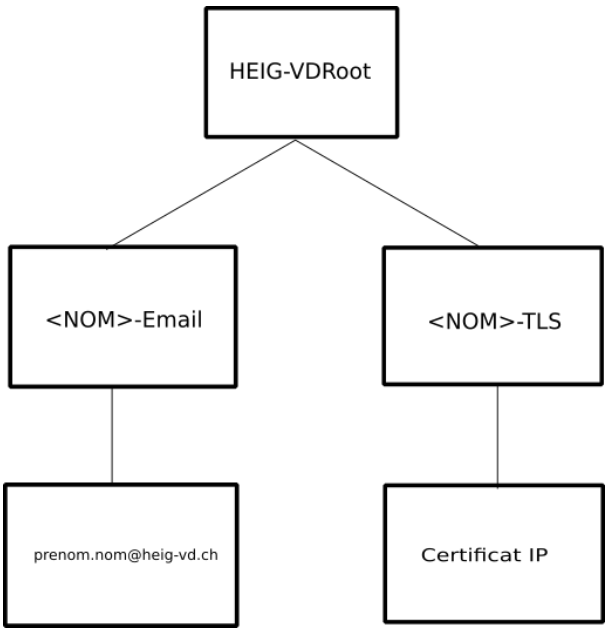
- Si vous utilisez des mots de passe pour sécuriser vos clefs utilisez le mot de passe **123456**.
- Le rendu est décrit dans la Section 8.
- N'oubliez pas de répondre aux questions de la Section 7.
- Le labo nécessite un VPN vers le réseau de l'école fonctionnel.
- Le labo nécessite de pouvoir utiliser les commandes `openssl`. L'idéal est de le faire depuis Linux mais il est aussi possible de la faire sous Windows.

1 Introduction

Le but de ce laboratoire est d'implémenter une PKI et de l'utiliser pour signer des emails ainsi que pour configurer une connexion HTTPS. Le laboratoire se divise en trois parties :

1. Configuration des CAs.
2. Signatures d'emails.
3. Configuration d'un serveur pour connexion HTTPS.

L'architecture de la PKI aura l'allure suivante avec <NOM>, votre nom de famille :



Le certificat racine HEIG-VDRoot est géré par nous mais vous devrez générer vous même tous les autres certificats.

2 Fichiers Fournis

Nous vous fournissons les fichiers suivants :

— Le certificat racine HEIG-VDRoot.

— Un template de fichier de configuration pour openSSL.

— Un mot de passe, une IP et un port afin de vous connecter à votre serveur TLS. (par email)

3 Commandes Utiles

Nous allons utiliser `openssl` afin de générer les certificats. Vous trouverez dans cette section toutes les commandes utiles à ce laboratoire.

Créer une nouvelle demande de signature de certificat (CSR). Afin d'obtenir un certificat, il est nécessaire de générer une demande de signature de certificat qui contient toutes les informations du certificat. En particulier, elle va contenir le **common name** du certificat (CN). Cette demande devra ensuite être signée par un CA. Vous pouvez obtenir la CSR à l'aide de la commande suivante :

```
openssl req -new -config <FICHIER_CONFIG> -out <OUTPUT_CSR>.csr -keyout <OUTPUT_KEY>.key
```

Le fichier de config pour un certificat intermédiaire vous est fourni mais est incomplet. A vous de le compléter.

Signer un nouveau certificat. Pour signer un certificat, il faut lui donner un numéro de série. OpenSSL gère ces numéros de série à l'aide d'un fichier qu'il lit et incrémente à chaque fois. Ce fichier doit être initialement créé avant de pouvoir appeler la commande qui signe un certificat. Pour créer une serial (initialisé à 1), utilisez la commande suivante :

```
echo 01 > <NOM_FICHIER>.srl
```

Les certificats sont aussi gérés dans une base de donnée qu'il faut initialiser de la manière suivante :

```
touch <NOM_FICHIER>.db
touch <NOM_FICHIER>.db.attr
```

Finalement, pour signer un certificat, nous utilisons la commande suivante :

```
openssl ca -config <FICHIER_CONFIG> -in <INPUT_CSR> \
   -out <OUTPUT_CERTIFICAT>.crt -extensions <EXTENSION_A_UTILISER>
```

Les différentes extensions à utiliser sont définies dans le fichier de config.

Pour obtenir un certificat auto-signé, ajoutez l'option `-selfsign`.

Visualiser un certificat. Pour visualiser un certificat dans le terminal, il faut utiliser la commande suivante :

```
openssl x509 -noout -text -in <CERTIFICAT>
```

Obtention d'une chaine de certificats. Il est parfois nécessaire d'utiliser une chaine de certificats. C'est typiquement le cas pour TLS, qui fourni les certificats intermédiaires en plus du certificat final afin de pouvoir tester facilement tous les certificats utilisés. Une chaine de certificats est simplement la **concaténation** des certificats. Vous pouvez utiliser la commande suivante pour obtenir une chaine de certificats :

```
cat <CERTIF_CLIENT> <CERTIF_INTERMEDIAIRE>  >  <CERTIF_CHAIN>.crt
```

Exporter un certificat en format PKCS#12. Le format PKCS#12 permet de combiner dans un même fichier le certificat et la clef privée associée. Il est typiquement utilisé pour signer des messages. Il va aussi parfois contenir une chaine de certificats (dans notre cas, le certificat racine ainsi que le certificat intermédiaire). Afin, d'exporter une clef et un certificat en PKCS#12, utilisez la commande suivante :

```
openssl pkcs12 -export -name <NOM> -inkey <CLEF> \
   -in <CERTIFICAT> -certfile <CHAIN> -out <FILE>.p12
```

La variable `<NOM>` sert uniquement à identifier le certificat de manière lisible.

4 Certificats Intermédiaires

La première étape de ce laboratoire va consister en l'obtention de certificats intermédiaires. Le premier est le certificat d'une CA qui sera utilisée pour signer des certificats utilisés pour les emails. Le second est le certificat d'une CA qui sera utilisée pour signer des certificats utilisés pour des connexions TLS.

1. Créez des CSR correspondant aux deux CA intermédiaires. Le common name doit correspondre au schéma, le pays doit être CH et l'organisation HEIG-VD.
2. Faites signer le certificat par notre CA racine en utilisant l'interface située à l'adresse `http://crypto.einet.ad.eivd.ch:5000`. **Très important** : ce serveur n'est accessible uniquement par VPN ou depuis le réseau de l'école.
3. Enregistrez le résultat dans un fichier `.crt` qui contiendra le certificat signé.
4. A la fin de cette étape, vous devez avoir un certificat ainsi qu'une clef privée pour chacune de deux CA.

5 Signatures des emails

Le but de cette étape va être d'envoyer un mail à `alexandre.duc@heig-vd.ch` signé par un certificat attestant de votre adresse email. Nous conseillons d'utiliser Thunderbird [1] pour envoyer l'email mais tout autre client mail acceptant le format S/MIME est autorisé. Pour ceci, nous allons effectuer les étapes suivantes :

1. Créez une CSR correspondant à votre certificat client. Attention au champ email !
2. Signez votre CSR avec votre CA intermédiaire utilisée pour les emails.
3. Configurez Thunderbird afin de le connecter à votre compte mail HEIG-VD.
4. Créez une chaine de certificats comprenant le certificat racine et votre certificat intermédiaire utilisé pour les emails.
5. Créez un bundle PKCS#12 comportant votre clef privée, votre certificat client, et la chaine de certificats du point précédent.
6. Vous pouvez gérer les certificats dans Thunderbird sous Préférences → avancé → Certificats → gérer les certificats
7. Ajoutez y le certificat racine puis votre certificat client.
8. Allez ensuite dans les paramètres de votre compte dans l'onglet sécurité pour choisir quelle clef vous souhaitez utiliser pour signer vos messages.
9. Envoyez le mail. N'oubliez pas de le signer !

6 Configuration d'un serveur TLS

Le but de cette étape va être de configurer un serveur nginx afin de visualiser de manière sûre une page en HTTPS. Nous allons utiliser TLS 1.2 et 1.3 pour effectuer la connexion HTTPS. [2] En particulier, la configuration du serveur sera très importante : aucune ciphersuite vulnérable n'est autorisée !

1. `https://www.thunderbird.net`
2. **Attention :** la connexion TLS 1.2 doit être aussi activée !

Vous avez reçu l'IP et le port de votre serveur ainsi que les credentials de connexion qui vous permettront de vous connecter vis SSH sur le serveur. **Très important :** le serveur n'est accessible uniquement via VPN (ou depuis le réseau de l'école). Le nom d'utilisateur sera `labo`. Vous pouvez donc vous connecter avec la commande :

```
ssh -p PORT labo@IP
```

Si vous souhaitez copier un fichier de votre machine vers la machine remote, vous pouvez utiliser SCP de la manière suivante :

```
scp -P PORT FICHIER_LOCAL labo@IP:CHEMIN_SERVEUR
```

Nous allons faire ceci à l'aide des étapes suivantes :

1. Créez une CSR correspondant à votre certificat client.
2. Signez votre CSR avec votre CA intermédiaire utilisée pour TLS.
3. Connectez-vous à votre serveur web.
4. Configurez nginx afin d'avoir une connexion HTTPS sûre. Le fichier à configurer se trouve sur le serveur à `/etc/nginx/sites-available/default`. Pour configurer TLS, vous pouvez vous aider du lien suivant : https://ssl-config.mozilla.org.[3]
5. Vous pouvez tester votre connexion depuis le browser.

7 Questions

Répondez dans votre rapport aux questions suivantes :

1. Lors de la génération de CSR, une pair de clefs est automatiquement générée. Il est aussi possible de séparer cela en deux étapes : générer tout d'abord les clefs et la CSR ensuite. Dans quel contexte est-ce que cette deuxième manière est préférable ?
2. Avez-vous chiffré la clef privée du serveur ? Justifiez.
3. Pourquoi devons nous transmettre une chaine de certificats dans les deux applications (email et TLS) ?
4. Que contiennent les fichiers .db des CAs ?
5. Comment avez-vous configuré nginx ? Donnez votre fichier de configuration.

8 Rendu

Vous serez évalués sur les points suivants :

1. Mail envoyé.
2. Configuration correcte du serveur TLS.
3. Petit rapport répondant aux questions de la Section 7.
4. Fichiers générés dans le laboratoire en suivant **l'arborescence suivante**[4] :

```
├── mail_CA
│   ├── config.conf
│   ├── <NOM>-Email.crt
│   ├── <NOM>-Email.csr
│   └── <NOM>-Email.key
├── mail_client
│   ├── prenom.nom@heig-vd.ch.crt
│   ├── prenom.nom@heig-vd.ch.csr
│   ├── prenom.nom@heig-vd.ch.key
```

3. Note : nous n'activerons pas l'OCSP stapling.
4. Les noms des fichiers peuvent changer mais pas ceux des dossiers.

```
│   └── prenom.nom@heig-vd.ch.p12
├── TLS_CA
│   ├── config.conf
│   ├── <NOM>-TLS.crt
│   ├── <NOM>-TLS.csr
│   └── <NOM>-TLS.key
└── TLS_client
    ├── IP.crt
    ├── IP.csr
    └── IP.key
```

Soumettez sur cyberlearn votre rapport ainsi que l'arborescence demandée.