Énoncé du TP Certificats électroniques [*] [†]

Isabelle Chrisment, Philippe Jaillon, Maryline Laurent,
Marwan Lazragi, Souha Masmoudi

Janvier 2020

Introduction

Une entreprise X a plusieurs implantations géographiques différentes qui sont par exemple Paris, Brest et Marseille. Ces différents sites sont connectés à Internet, ce qui permet à tout le personnel d'échanger facilement des informations. Néanmoins, tous les échanges qui ont un caractère officiel ne passent pas par ce réseau et se font sous forme papier en utilisant le courrier postal. L'entreprise trouve que c'est la seule façon suffisamment sécurisée pour diffuser des données confidentielles. Par contre, elle trouve que cette méthode n'est pas suffisamment performante et qu'elle est trop coûteuse.

Le nouveau responsable d'information de l'entreprise propose d'utiliser le World Wide Web pour résoudre ces problèmes de coût et d'efficacité. Aux remarques que sa direction lui fait concernant la confidentialité des échanges et l'authentification des personnels, il suggère d'utiliser les certificats électroniques qui sont d'après lui une solution souple et robuste lorsqu'ils sont correctement mis en œuvre.

Nous allons voir dans ce TP comment créer les certificats électroniques, comment les utiliser et en découvrir tout l'intérêt.

Environnement

Pour simuler de manière réaliste notre environnemt de travail, vous disposez dans le `labtainer certificate` de 4 machines connectées entre elles. La machine `www` abritera le serveur web de l'entreprise, la machine `CA` sera dédiée à l'autorité de certification (toutes les opérations de création et de signature de certificats y seront réalisées). Les machines `brest` et `marseille` seront des machines utilisées respectivement par les utilisateurs `goodguy` et `badboy`. Les machines sont configurées pour utiliser les services d'un DNS et peuvent donc être contactées directement par leur nom.

Démarrage du TP

Après avoir démarré votre machine virtuelle *Labtainers*, depuis le répertoire courant (`~/labtainer/labtainer-student`), exécutez la commande suivante :

```
$ labtainer certificate
```

[*] Réalisé dans le cadre du MOOC sécurité des réseaux financé par le projet ANR FLIRT et la fondation DRAHI.

1 Création du certificat de l'autorité de certification

Avant de créer les certificats, l'entreprise X doit tout d'abord se choisir une autorité de certification qui va par la suite lui délivrer les différents certificats dont elle aura besoin. Pour des raisons qui lui sont propres, elle a décidé de créer et de gérer sa propre autorité de certification.

Une autorité de certification (CA) doit disposer d'un certificat particulier qui s'appelle un *certificat racine*. Ce certificat à la particularité d'être signé par sa propre clef privée, de cette manière, tout le monde pourra vérifier qu'il a bien été signé par le possesseur de la clef privée associée à la clef publique disponible dans le certificat.

Si la clé publique peut être largement diffusée, la clé privée est au contraire confidentielle et doit être très bien protégée : il s'agit de l'élément sur lequel repose toute la sécurité du système. Si un tiers prend connaissance de la clef privée de la CA, il pourra alors générer de *vrais faux* certificats (des certificats reconnus valides par toutes les personnes faisant confiance à cette CA, mais qu'elle n'aura pas elle-même signés).

Notre première tâche va donc consister à générer la paire de clefs (publique/privée) de l'autorité de certification. Pour ce faire, sélectionnez la console de la machine CA, vérifiez que vous êtes bien dans le *home directory* de l'utilisateur *authority* (/home/authority).

```
CA$ openssl genrsa -des3 -out private/ca.key.pem 2048
```

- L'option `-des3`, signifie que la paire de clés est protégée par un chiffrement utilisant l'algorithme **DES3**. Pour prendre connaissance de la clef, il est nécessaire de connaître la « passphrase » (mot de passe long qui peut même contenir des espaces) qui la protège.
- L'option `-out` permet de spécifier le nom du fichier qui contient la paire de clés.
- Le dernier paramètre est la taille de la clé en bits (2048 bits dans notre exemple).

Vous pouvez afficher les différents éléments de la clé générée en tapant la commande suivante :

```
CA$ openssl rsa -in private/ca.key.pem -text
```

Q1: Est-il possible d'accéder à cette clé ? Expliquer.

a) Oui

b) Non

Maintenant que nous disposons d'un couple de clefs, nous allons générer le certificat de notre autorité de certification. Il va s'agir d'un certificat X.509 auto-signé (une racine de certification). Pour ce faire, nous allons signer les informations concernant cette autorité de certification associée à sa clef publique en utilisant la clé privée. Ce certificat aura une durée de validité de 5 ans (1825 jours).

```
CA$ openssl req -config openssl.cnf -new -x509 -days 1825
-extensions v3_ca -key private/ca.key.pem -out ca.crt.pem
```

- l'option `-config` permet de désigner le fichier de configuration à utiliser (pour le TP nous mettons à votre disposition un fichier de configuration adapté à notre environnement : openssl.cnf).
- Le paramètre `-new` combiné avec **`x509`** indique que l'on veut un certificat X.509 auto-signé.
- Le paramètre `-days` indique la durée de validité du certificat (en jours).
- Le paramètre `-key` spécifie la clé RSA à utiliser.
- Le paramètre `-extensions` indique quelle section du fichier de configuration spécifie les extensions X509 qui doivent être ajoutées au certificat (la principale étant que ce certificat est celui d'une autorité de certification).

- Le paramètre `-out` définit le nom du fichier contenant le certificat qui sera généré.

Q2: Pourquoi faut-il rentrer une *passphrase* lors de l'exécution de cette commande ?

a) Pour prouver que la clé privée utilisée nous appartient.

b) Pour protéger le certificat généré.

Lors de l'exécution de la commande, nous sommes interrogés sur les valeurs que doivent prendre les différents champs de ce qui s'appelle le *distinguished name* (ou DN) : le nom qui sera associé au certificat. Ces informations doivent être rentrées avec soin. Une fois que le certificat de notre autorité de certification sera signé, il ne sera plus possible de les modifier et tous les certificats issus de cette CA y feront référence.

```
Country Name (2 letter code) [FR]:
Sate or Province Name (full name) [FRANCE]:
Organization Name (eg, company) [AA MOOC-IMT]:
Organizational Unit Name (eg, section) [NETWORK SECURITY]:
Common Name (e.g. server FQDN or YOUR name) []: MOOC_CA
Email Address []: CA@mooc.imt
```

2 Création de certificats

Maintenant que nous disposons des clefs et du certificat de notre autorité de certification, nous allons pouvoir délivrer des certificats aux différents intervenants de l'entreprise.

Création du certificat du serveur

Nous allons commencer par créer un certificat pour le serveur web de l'entreprise : `www.mooc.imt`. La première étape est la génération d'une paire de clés qui lui est propre :

```
CA$ openssl genrsa -des3 -out server.key.pem 1024
```

Q3: Que signifie le paramètre 1024 dans la commande précédente ?

a) Durée de validité de la paire de clés.

b) Nombre de bits de la paire de clés.

Maintenant que nous disposons de cette clé, nous allons générer une demande de certification (Certificate Signing Request - CSR). Cette opération est faite par le gestionnaire du serveur web, tout comme l'étape précédente, la clef privée du serveur n'ayant pas besoin d'être connue de l'opérateur de certification.

```
CA$ openssl req -config openssl.cnf -new -key server.key.pem
                        -out server.req.pem
```

NB : la valeur du `cn` renseignée lors de la création du certificat doit être le nom du serveur pour que nos navigateurs web acceptent d'établir des connexions sécurisées : Common name (CN) = www.mooc.imt

Après avoir transmis la demande de certification à la CA (nous changeons juste de casquette puisque la *CSR* est déjà disponible sur la machine CA), il ne reste plus maintenant à l'opérateur de certification qu'à valider le contenu du certificat, à y ajouter les extensions nécessaires à la fonction de serveur (vous pouvez consulter le fichier de configuration, section *server*) et à signer le certificat pour le serveur `www.mooc.imt`.

```
CA$ openssl ca -config openssl.cnf -in server.req.pem
        -out server.crt.pem -extensions server
```

Q4: Quels sont les éléments que l'on peut retrouver dans le certificat électronique d'une machine ?

a) Le nom, URL de la machine.

b) La clé privée de la machine.

c) La signature d'un tiers de confiance (autorité de certification).

d) La période de validité du certificat.

3 Connexion sécurisée à un serveur web

Pour respecter son cahier des charges, l'entreprise X doit sécuriser les échanges avec son serveur web (il s'agit d'un serveur web *apache*) à l'aide du protocole HTTPS. Ce serveur web doit donc pouvoir utiliser le certificat que nous venons de créer.

Pour ce faire, recopiez le certificat du serveur et sa clé privée ainsi que le certificat de l'autorité de certification qui se trouve dans la machine CA sur la machine abritant le serveur web [1].

```
www$ sudo scp authority@CA:server.key.pem /etc/apache2/certs/server.key.pem

www$ sudo scp authority@CA:server.crt.pem /etc/apache2/certs/server.crt.pem

    www$ sudo scp authority@CA:ca.crt.pem /etc/apache2/certs/ca.crt.pem
```

Le serveur web a été pré-configuré [2] pour utiliser le protocole HTTPS et prendre la clef privée et le certificat dont il a besoin pour prouver son identité dans le répertoire `/etc/apache2/certs/`. Vérifiez que ces deux documents y sont bien présents.

Si tout est correct, lancez le service web sur la machine `www` :

```
www$ sudo systemctl start apache2
```

Lorsque le serveur vous le demande, fournissez le mot de passe pour déverrouiller la clef privée du serveur web.

Pour vérifier et tester le bon fonctionnement des communications sécurisées, sur une des deux machines `brest` ou `marseille`, lancez Firefox depuis la console :

```
brest$ firefox &
```

et connectez-vous en https au serveur web de l'entreprise :
`https://www.mooc.imt/`

Normalement, si la configuration de votre serveur *apache* est correcte, votre navigateur web doit vous afficher un certain nombre d'alertes et vous empêcher dans un premier temps d'accéder au serveur.

Q5: A quoi correspondent ces alertes ?

a) Le certificat présenté par le serveur est un faux ;

b) Le certificat n'est pas fait pour ça ;

c) Le certificat ne peut pas être vérifié.

1. La machine CA dispose d'un serveur ssh. Le mot de passe de l'utilisateur `authority` est `authority`
2. La configuration utilisée se trouve dans le fichier `/etc/apache2/sites-available/default-ssl.conf`

Pour que la confiance puisse s'établir entre le client et le serveur, le client doit posséder une copie du certificat de l'autorité de certification qui a signé le certificat qui lui est présenté par le serveur. Nous devons donc faire une copie du certificat de la CA sur la machine du client (ici `brest`) et importer ce certificat dans notre navigateur web.

```
brest$ scp authority@CA:ca.crt.pem .
```

Sur Firefox, choisissez dans *Menu* > *Preferences* > *Privacy & Security* > *Certificates* > *View certificates* > *Authorities* > *Import* et sélectionnez le certificat de la CA que vous venez recopier. Vérifiez que le certificat que vous allez importer est bien celui de la CA et indiquez sur la boite de dialogue qui vous est présentée que vous faites confiance à cette CA pour vérifier l'identité de serveurs web et de mails utilisateurs. Validez l'importation du certificat et renouvelez votre requête au serveur. Vous pouvez répéter l'ensemble de ces opérations sur la machine `marseille`.

Q6: Qu'en déduisez-vous ? [1 bonne réponse]

a) Je n'en sais rien ;

b) Mon navigateur est à même de vérifier le certificat du serveur car il possède une racine de certification qui le lui permet ;

c) Mon navigateur est à même de vérifier le certificat du serveur car il possède le certificat du serveur ;

d) Mon navigateur s'est mis dans un mode plus tolérant et accepte tous les certificats des serveurs.

4 Authentification par certificat

Toujours pour satisfaire le cahier des charges de l'entreprise X, nous devons fournir à chaque utilisateur un certificat qui doit lui permettre de s'authentifier sur le serveur web de l'entreprise.

Les certificats clients

De la même manière que lors de la création du certificat pour le serveur (voir la section 2), créez un certificat pour chacun des utilisateurs humains `goodguy` et `badboy` (créez une clef, une CSR correctement renseignée pour l'utilisateur concerné, puis le certificat lui même en prenant soin de bien positionner l'option `-extensions client`).

Pour que les certificats que nous venons de créer permettent aux utilisateurs de s'authentifier sur le serveur web de l'entreprise, le certificat et sa clef privée associée doivent être disponibles pour leur navigateur web (Firefox).

Pour importer un certificat et sa clef privée dans un navigateur web, il est nécessaire de stocker ces différents éléments dans un conteneur sécurisé : un fichier au format PKCS#12. Ce format de fichier est un format d'échange standardisé par RSA. Il est utilisé pour les échanges de tout le matériel relatif à un certificat (clé privée, clé publique, certificat et chaîne de certification). Il permet de simplifier et de sécuriser les opérations d'importation.

La création du fichier PKCS#12 se fait avec la commande suivante :

```
CA$ openssl pkcs12 -export -in goodguy.crt.pem -inkey goodguy.key.pem
      -out goodguy.p12 -name "Good Guy" -certfile ca.crt.pem
```

- l'option `-export` spécifie le type d'opération à faire : exporter le matériel au format PKCS#12 ;
- l'option `-in` spécifie le nom du fichier contenant le certificat (`goodguy`.crt.pem)
- l'option `-inkey` désigne le fichier contenant la clef privée associée (`goodguy`.key.pem) ;

- l'option `-out` spécifie le nom du fichier ou seront stockées les données au format PKCS#12 (`goodguy`.p12) ;
- l'option `-name` est un commentaire présenté à l'utilisateur lors de l'importation ;
- l'option `-certfile` permet d'inclure des certificats supplémentaires. Ici nous avons ajouté le certificat de l'autorité de certification qui a signé le certificat utilisateur.

Répétez l'opération pour l'utilisateur `badboy` puis transférez les fichiers `.p12` sur les machines utilisateur : `badboy`.p12 sur la machine `marseille` et `goodguy`.p12 sur la machine `brest`. Le contenu des fichiers PKCS#12 peut être affiché avec la commande suivante :

```
brest$ openssl pkcs12 -in goodguy.p12
```

Q7: La clé privée est-elle en clair ?

a) Oui

b) Non

Connexion au serveur web

Nous allons maintenant modifier la configuration du serveur *apache* de manière à ce qu'il soit capable d'authentifier tout utilisateur qui se présenterait avec un certificat issu de notre CA.

Modifiez le fichier de configuration `/etc/apache2/sites-enabled/default-ssl.conf`. Vous devez décommenter les lignes étiquetées `"# 1 -->"`, c'est-à-dire supprimer les `"# 1 -->"` et ne pas laisser de marge à gauche pour cette ligne décommentée. Cette opération modifie les propriétés de votre serveur web de la manière suivante :

- la directive `SSLVerifyClient` impose au client de fournir un certificat et la directive `SSLVerifyDepth` impose qu'il soit validable ;
- les directives `SSLCertificateChainFile` et `SSLCACertificateFile` précisent quels sont les certificats des CA qui doivent être utilisés pour la validation des certificats clients ;
- `SSLUserName` permet d'extraire une identité d'un des champs du certificat présenté, cette information est logguée et peut être utilisée par exemple lorsque l'on utilise des groupes ;
- la directive `Require` appliquée à l'objet `<directory />` contrôle l'accès aux documents disponibles sur le serveur. Dans notre cas, l'accès est autorisé si et seulement si le champ *Organisation* (`O`) du *subject* du certificat présenté a pour valeur `AA MOOC IMT`.

Une fois le fichier de configuration modifié, vous pouvez relancer le serveur *apache* :

```
www$ sudo systemctl restart apache2
```

Vous pouvez observer le fonctionnement du server web en consultant les fichiers de log qui se trouvent dans le répertoire `/var/log/apache`. L'usage de la commande `tail` et de son option `-f` permet de voir en *temps réel* les requêtes faites au serveur.

```
www$ tail -f /var/log/apache/ssl_access.log
```

Sur les machines `brest`, puis `marseille`, utilisez de nouveau Firefox pour vous connecter au serveur web.

Q8: Que constatez-vous ?

a) Je n'arrive pas à me connecter ;

b) J'arrive à me connecter ;

c) Je ne sais pas comment spécifier mon identité.

Sur chacune des machines `brest` et `marseille`, importez le fichier `.p12` dans votre navigateur web. Vérifiez la validité de votre certificat utilisateur (information que l'on peut consulter en double-cliquant sur le certificat). Si votre certificat n'est pas valide, c'est que vous avez sans doute oublié d'indiquer que le certificat de la CA pouvait être utilisé pour valider des certificats serveurs et des certificats utilisateurs (emails).

Depuis, successivement `brest` puis `marseille`, consultez de nouveau l'url `https://www.mooc.imt/`

Q9: Que constatez-vous ?

a) Je n'arrive pas à me connecter ;

b) J'arrive à me connecter ;

c) Je ne sais pas comment spécifier mon identité.

5 Gestion de la validité des certificats

Lorsque l'autorité de certification délivre un certificat, elle lui associe une période de validité : sa date de création et sa date de fin de validité. Ce certificat sera donc considéré comme valide pendant toute cette période. Mais la situation du propriétaire du certificat peut changer. Par exemple, s'il s'agit d'une personne, elle a pu quitter l'entreprise, changer d'affectation et certaines actions qui lui étaient autorisées n'ont alors plus de raison d'être, la clef privée du certificat a pu être compromise... Il est donc nécessaire de disposer d'un moyen pour invalider un certificat, indépendamment des informations contenues dans le certificat lui même.

5.1 Liste de certificats révoqués (CRL)

Le premier moyen qui a été choisi est ce que l'on appelle les listes de révocations (*Certificate Revocation List* - CRL en anglais). Le fonctionnement est le suivant : lors d'une demande de révocation, le numéro de série du certificat concerné est ajouté à la liste des certificats déjà révoqués et cette liste est signée par le gestionnaire de la PKI de manière à ce que les utilisateurs puissent y faire confiance (révoquer un certificat est une opération qui doit être faite avec autant de sérieux que la signature d'un certificat).

Avec `openssl`, la révocation d'un certificat se fait en deux étapes :

1. la révocation du certificat lui même. Le fait que le certificat soit révoqué est consigné dans le fichier `index.txt` de votre CA. Pour le certificat concerné, on y trouve la date de révocation et la raison de cette révocation.

2. La génération de la liste de révocations. `openssl` utilise l'ensemble des renseignements disponibles dans le fichier `index.txt` pour établir la liste de révocations. Cette liste est alors signée par la CA.

Nous allons maintenant révoquer le certificat de l'utilisateur `badboy` puis générer une liste de révocations. Ces opérations sont réalisées sur la machine CA.

Révocation d'un certificat

```
CA$ openssl ca -config openssl.cnf -revoke badboy.crt.pem
                -crl_reason keyCompromise
```

Génération de la liste de révocations

```
CA$ openssl ca -config openssl.cnf -gencrl -out crl.pem
```

Vous pouvez consulter le contenu cette liste révocations avec la commande suivante :

```
CA$ openssl crl -text -in crl.pem
```

Q10: A quoi sert un certificat révoqué ?

a) A chiffrer les messages.

b) A vérifier des signatures.

c) Ni à chiffrer des messages, ni à vérifier des signatures.

Sur la machine `marseille`, connectez-vous avec Firefox au serveur web.

Q11: Que constatez-vous ?

a) Ça n'a rien changé ;

b) L'utilisateur `badboy` ne peut plus se connecter ;

c) Révoquer un certificat n'est pas immédiatement pris en compte.

5.2 Serveur web et liste de révocations

Pour que notre serveur web puisse tenir compte des informations de révocation, nous devons y recopier la liste de révocations et modifier le fichier de configuration du serveur *apache* pour qu'elle soit prise en compte.

Modifiez le fichier de configuration `/etc/apache2/sites-enabled/default-ssl.conf`. Vous devez décommenter les lignes étiquetées `"# 2 -->"`. Cette opération modifie les propriétés de votre serveur web de la manière suivante :

- la directive `SSLCARevocationFile` désigne la liste de révocations à prendre en compte (vérifiez que vous avez bien recopié le fichier crl.pem à l'emplacement indiqué) ;
- la directive `SSLCARevocationCheck` spécifie le mode de vérification que le serveur doit mettre en œuvre. Ici, il s'agit du mode `chain` qui indique que pour chaque certificat présenté, l'intégralité de la chaîne de certification doit être vérifiée.

Relancez le serveur *apache* :

```
www$ sudo systemctl restart apache2
```

Sur la machine `marseille`, puis sur la machine `brest`, connectez-vous avec Firefox au serveur web (firefox gérant un cache des pages déjà lues, forcez le *reload* à l'aide du bouton prévu à cet effet).

Q12: Que constatez-vous ?

a) Ça n'a rien changé ;

b) L'utilisateur `badboy` ne peut plus se connecter ;

c) L'utilisateur `badboy` peut toujours se connecter ;

d) L'utilisateur `goodguy` ne peut plus se connecter ;

e) L'utilisateur `goodguy` peut toujours se connecter ;

f) Il est nécessaire de disposer d'une copie de la liste de révocations.

5.3 OCSP *(Cette section est optionnelle)*

Nous venons de voir, que pour que la révocation d'un certificat soit prise en compte, un certain nombre d'opérations sont nécessaires :

- il faut interroger régulièrement la CA pour savoir si des mises à jour de la liste de révocations ont été faites ;
- faire une copie de la liste de révocations sur son serveur web ;
- relancer le serveur web pour que la nouvelle liste soit prise en compte.

Toutes ces opérations entraînent des délais entre le moment où le certificat est révoqué et le moment où notre serveur web peut en tenir compte, ce qui peut poser des problèmes de sécurité dans certaines situations.

C'est pour répondre à ce problème que le protocole OCSP (Online Certifcate Status Protocol) a été proposé. L'idée est la suivante : toute autorité de certification peut mettre en œuvre un service de validation en ligne des certificats qu'elle a signés. Dans chacun des certificats est indiquée l'URL de ce service de validation. Il suffit d'interroger ce service pour vérifier la validité du certificat qui est présenté. Pour s'assurer de la validité de cette vérification, les informations échangées sont signées par l'autorité de certification elle-même ou par un certificat *habilité* (signé par cette CA et disposant de l'extension d'usage `OCSPSigning`).

Si vous étudiez les informations contenues dans les certificats, vous pourrez constater qu'ils disposent de l'attribut `authorityInfoAccess` qui a pour valeur `OCSP;URI:http://ocsp.mooc.imt:8888`. Cet attribut indique simplement comment et où contacter le serveur OCSP géré par notre CA.

Pour mettre en œuvre un service OCSP, la première chose à faire est de lui créer un certificat. Ce certificat lui permettra d'indiquer aux utilisateurs qu'il est légitime à signer des réponses OCSP pour notre CA (`extendedKeyUsage=OCSPSigning`). Comme pour notre serveur web, la création du certificat pour le serveur OCSP se fait en 3 étapes :

- création d'une clé RSA,
- génération de la *Certificate Signing Request* en précisant bien que le champ `CN` a pour valeur `ocsp.mooc.imt` [3],
- et enfin génération et signature d'un certificat possédant les bonnes extensions(`-extensions ocsp_ext`).

```
CA$ openssl genrsa -des3 -out ocsp.key.pem 1024

CA$ openssl req -config openssl.cnf -new -key ocsp.key.pem
                    -out ocsp.req.pem

CA$ openssl ca -config openssl.cnf -in ocsp.req.pem
      -out ocsp.crt.pem -extensions ocsp_ext
```

Nous allons maintenant lancer un serveur OCSP sur la machine qui héberge votre CA.

Pour revenir en arrière, nous allons faire comme si le certificat de `badboy` n'avait pas été révoqué en modifiant le fichier `index.txt` :

- sur la ligne concernant `badboy`, nous allons remplacer le `R` au début de la ligne par `V` (R=Revoked, V=Valid) ;
- puis supprimez le 3ème champ (il s'agit de la date et de la raison de la révocation) ;
- enfin, enregistrez le fichier.

3. Pour le serveur DNS que nous utilisons, la machine `ocsp.mooc.imt` a été définie comme étant un alias (un autre nom) de la machine `ca.mooc.imt`

Nous devons configurer votre serveur *apache* de manière à ce que, maintenant, il utilise OCSP pour vérifier la validité des certificats des clients. Pour ce faire nous devons commenter les deux lignes concernant l'utilisation des listes de révocations (elles sont étiquetées `"# <-- 2"`) et décommenter la ligne concernant OCSP (étiquetée `"# 3 -->"`).

La commande `ocsp` de `openssl` permet de tester le fonctionnement de ce protocole. Lancez le serveur OCSP sur la machine CA.

```
ca$ openssl ocsp -port 8888 -index index.txt -CA ca.crt.pem
    -rsigner ocsp.crt.pem -rkey ocsp.key.pem -req_text
```

Relancez le serveur web pour que la nouvelle configuration soit prise en compte :

```
www$ sudo systemctl restart apache2
```

Connectez-vous au serveur web avec Firefox et vérifiez que `goodguy` et `badboy` ont bien accès tous les deux au serveur.

Révoquez le certificat de `badboy` sur la CA et relancez le serveur OCSP.

```
CA$ openssl ca -config openssl.cnf -revoke badboy.crt.pem
                -crl_reason keyCompromise
```

Reconnectez-vous au serveur web à partir de la machine `brest` avec l'identité `goodguy`, puis à partir de la machine `marseille` avec l'identité `badboy`.

Q13: Que constatez-vous ?

Sur la CA, faites une copie du fichier index.txt, puis révoquez le certificat du serveur web, puis relancez le serveur OCSP.

```
CA$ openssl ca -config openssl.cnf -revoke server.crt.pem
                -crl_reason keyCompromise
```

Depuis la machine `brest`, re-connectez-vous au serveur web.

Q14: Que constatez-vous ?

5.4 OCSP Stapling

A l'échelle de l'Internet, la mise en œuvre du protocole OCSP génère un très grand nombre de requêtes. Potentiellement, chaque fois qu'un client se connecte à un serveur, il peut vouloir vérifier la validité du certificat qui lui est présenté. Pour limiter ce surcoût, les navigateurs web mettent en cache les réponses OCSP et tant que ces données ne sont pas invalidées, ils ne sont pas informés des certificats révoqués. De plus, la vérification de la validité des certificats prend du temps au client et est considérée comme néfaste à l'expérience utilisateur. Pour tenter de régler ces problèmes, une nouvelle méthode a été proposée : l'agrafage OCSP (*OCSP-stapling*). L'idée est la suivante : à intervalle régulier, le serveur demande au serveur OCSP de sa CA de lui fournir la confirmation de la validité de son propre certificat. Il peut alors la transmettre avec son certificat lors de la phase de négociation de la connexion TLS. De cette manière, le coût pour obtenir la validation du certificat n'est plus à la charge du client (mais du serveur), et le nombre de requêtes de validation OCSP est très fortement réduit (seuls les serveurs font des demandes pour leur propre certificat et à une fréquence assez faible, toutes les heures par exemple). Si cette nouvelle méthode limite le trafic vers les serveurs OCSP, elle ne permet pas d'obtenir en temps réel les informations de validité du certificat. En effet, la réponse OCSP est datée par le serveur puis agrafée au certificat pendant un certain temps, période pendant laquelle il a pu être révoqué.

La mise en œuvre de l'*OCSP Stapling* est très simple dans *apache*. Elle consiste à activer la directive `SSLUseStapling` (étiquetée `"# <-- 4"`) dans le fichier de configuration et à ajuster, si

cela s'avère nécessaire les directives concernant la gestion du cache (directives `SSLStaplingStandardCacheTimeout` et `SSLStaplingErrorCacheTimeout`). Une fois les modifications réalisées et enregistrées, il suffit de relancer le serveur web qui se connectera au serveur OCSP de sa CA lors de la première connexion.

Vous pouvez vérifier que l'agrafage OCSP fonctionne en lançant la commande suivante sur `brest` :

```
brest$ openssl s_client -connect www.mooc.imt:443 -CAfile ca.crt.pem -status
```

Dans les informations liées à la négociation de connexion, vous trouverez une copie de la réponse OCSP faite au serveur et le statut de cette réponse.

Nous allons maintenant procéder à quelques tests :

pour ce faire, restaurez le fichier `index.txt` sauvegardé précédemment (où les serveurs et les clients ont tous des certificats valides). Relancez le serveur OCSP, le serveur web et firefox sur `brest` ou `marseille`. Si tout est configuré convenablement, vous accédez sans erreur à www.mooc.imt/.

Arrêtez maintenant le serveur OCSP sur ca.mooc.imt , puis relancez firefox sur `brest` ou `marseille`.

Q15: Firefox nous indique que la connexion est sécurisée

a) car le certificat serveur n'a pas été révoqué ;

b) la réponse OCSP est valide et est toujours transmise par le serveur.

Relancez firefox et renouvelez les tests précédents au bout de quelques minutes (attendre au moins 3 mn)

Q16: Firefox nous indique que la connexion n'est pas sécurisée

a) il n'a pas pu validé le certificat auprès du serveur OCSP ;

b) le certificat présenté par le serveur web ne contient pas d'agrafage OCSP ;

c) le certificat présenté par le serveur web ne contient pas d'agrafage OCSP et firefox n'a pas pu valider le certificat auprès du serveur OCSP.

Révoquez le certificat du serveur web puis relancez le serveur OCSP.

On peut forcer l'usage systématique de OCSP dans Firefox en accédant la page de configuration (`about:config`) et en activant la directive suivante :

- `security.OCSP.require` : *the security.OCSP.require to false (the default) means that the OCSP it is not forced. Check the* `security.OCSP.enabled` *it is 1 (active, enabled) (0, zero, is disabled).*

Quoi qu'il en soit, les caches, que ce soit dans firefox ou que ce soit avec le serveur web, semble se comporter de manière assez bizarre... Pour constater convenablement les comportements recherchés, le plus simple reste de stopper les services et les programmes puis de les relancer en fonction de ce que l'on compte observer.

Q17: Dernière question : quel intérêt voyez-vous à l'utilisation du protocole OCSP ?