TP SSL (OpenSSL)

1.OpenSSL, c'est quoi ?

1.1 Protocole SSL

Le terme SSL est un acronyme pour *Secure Socket Layer* qui est un protocole (en fait un ensemble de protocoles) qui a été développé par la société *Netscape Communication Corporation* pour permettre de la communication sécurisée en mode client/serveur pour des application réseaux utilisant TCP/IP. Le protocole TLS (*Transport Layer Security*) est une évolution de SSL réalisé par l'IETF et qui sert de base à HTTPS par exemple.

Le protocole SSL est entre la couche TCP/IP et une application utilisant TCP. Le principe générale d'un protocole de type SSL est qu'il se passe en deux temps :

1. Une poignée de mains : c'est une étape durant laquelle le client et le serveur s'identifient, se mettent d'accord sur le type du système de chiffrement et les clefs qui seront utilisés lors du reste de la communication.
2. La phase de communication : les données sont alors échangées en format compressées et chiffrées et signées.

1.2 OpenSSL

La bibliothèque OpenSSL est une implantation libre des protocoles SSL et TSL qui donne accés à :

• une bibliothèque de fonctionnalité écrite en C permettant de réaliser des applications client/serveur sécurisées s'appuyant sur SSL/TSL,

• un ensemble d'exécutables en commande en ligne permettant :

− la forge de clef RSA, DSA (pour les signature)

− la création de certificat X509 (identification)

− le calcul d'empreinte (MD5, SHA, RIPEMD160, ...)

− le chiffrement et le déchiffrement (RSA, DES, IDEA, RC2, RC4, Blowfish, ...)

− la réalisation de de tests de clients et serveurs SSL/TSL

− la signature et le chiffrement de courriers (S/MIME).

A tout instant vous pouvez avoir une vue sur l'ensemble des fonctionnalités de OpenSSL à l'aide des pages de manuel (man openssl).

La syntaxe générale pour l'utilsation en mode shell des fonctionnalités OpenSSL est la suivante

:

$ openssl <commande> <options>

le $ représente le prompt du shell.

2.Opérations de base avec OpenSSL

Vous pouvez utiliser les fonctionnalités suivantes :

• $ openssl enc <-algo> -in <claire.txt> -out <chiffre.enc> : pour le chiffrement de claire.txt avec l'algorithme spécifié (openssl enc --help pour avoir la liste des possibilités ou bien openssl list-cipher-commands) dans un fichier chiffre.enc.

• $ openssl enc <-algo> -in <chiffre> -d -out <claire> : pour le déchiffrement.

• $ openssl genrsa -out <fichier_rsa.priv> <size> : génére la clé privé RSA de taille size. les valeurs possible pour size sont : 512, 1024, etc.

• $ openssl rsa -in <fichier_rsa.priv> -des3 -out <fichier.pem> : chiffre la clef privé RSA avec l'algorithme DES3. Vous pouvez utiliser DES, 3DES, IDEA, etc.

• $ openssl rsa -in <fichier_rsa.priv> -pubout -out <fichier_rsa.pub> : stocke la partie publique dans un fichier à part (création de de la clé publique associée à la clef privée dans le fichier fichier.pem).

• $ openssl dgst <-algo> -out <sortie> <entrée> : pour hacher un fichier. L'option <-algo> est le choix de l'algorithme de hachage (sha, sha1, dss1, md2,md4, md5, ripemd160).

• $ openssl rand -out <clé.key> <nombre_bits> : pour générer un nombre aléatoire de taille nombre_bits (utiliser l'option -base 64 pour la lisibilité).

• $ openssl aes-256-cbc -in <claire.txt> -out <chiffre.enc> -e -k <clé.key>: pour chiffrer un fichier avec l'AES.

• $ openssl rsautl -encrypt -inkey <rsa.pub> -in <clair.txt> -out <chiffre.enc> : chiffrer fichier.txt avec la RSA en utilisant la clef publique rsa.pub.

• $ openssl rsautl -decrypt -inkey <rsa.priv> -in <chiffre.enc> -out <fihcier.txt> : pour déchiffrer le fichier fic.dec.

• $ openssl rsautl -sign -inkey <ras.priv> -in <fichier.txt> -out <fic.sig> : pour générer une signature.

• $ openssl rsautl -verify -pubin -inkey <rsa.pub> -in fic fic.sig : pour verifier une signature.

3.Travail demandé

1ère partie : Chiffrement symétrique

La commande qui vous permet d'utiliser le chiffrement symétrique est la commande *enc (en tapant **enc –help** vous aurez toutes les options associées)*

• $ openssl enc <-algo> -in <claire.txt> -out <chiffre.enc> : pour le chiffrement de claire.txt avec l'algorithme spécifié (openssl enc --help pour avoir la liste des possibilités ou bien openssl list-cipher-commands) dans un fichier chiffre.enc.

• $ openssl enc <-algo> -in <chiffre> -d -out <claire> : pour le déchiffrement.

Question 1 :

Soit un fichier donné ***fic_nom_eleve*** (choisissez un fichier qui contient des données textuelles). Ecrire la commande qui permet de le chiffrer et produit ainsi un fichier ***fic_nom_eleve.enc*** On peut alors vérifier que le message ***fic_nom_eleve.enc*** est bien inintelligible...

Ce même message est transmis à votre camarade qui pourra également le déchiffrer.

On peut alors vérifier que le message ainsi déchiffré est bien identique à ***fic_nom_eleve*** ...

Il vous est bien sûr de comprendre l'ensemble des manipulations associées.

Il vous est demandé de diversifier les algorithmes. D'utiliser également l'option –a pour produire un fichier chiffré lisible donc codé en base64 (lisible ne veut pas dire en clair).

2ème partie : Chiffrement asymétrique

Commandes nécessaires :

• $ openssl genrsa -out <fichier_rsa.priv> <size> : génére la clé privé RSA de taille size. les valeurs possible pour size sont : 512, 1024, etc.

• $ openssl rsa -in <fichier_rsa.priv> -des3 -out <fichier.pem> : chiffre la clef privé RSA avec l'algorithme DES3. Vous pouvez utiliser DES, 3DES, IDEA, etc.

• $ openssl rsa -in <fichier_rsa.priv> -pubout -out <fichier_rsa.pub> : stocke la partie publique dans un fichier à part (création de de la clé publique associée à la clef privée dans le fichier fichier.pem).

• $ openssl rsautl -encrypt –inkey –pubin <rsa.pub> -in <clair.txt> -out <chiffre.enc> : chiffrer fichier.txt avec la RSA en utilisant la clef publique rsa.pub.

• $ openssl rsautl -decrypt -inkey <rsa.priv> -in <chiffre.enc> -out <fihcier.txt> : pour déchiffrer le fichier fic.dec.

1.Forgez vos clefs RSA 512.

2. Chiffrer le petit fichier en RSA.

3. Envoyez votre clé publique à un voisin. Celui-ci vous enverra la sienne. Générer un petit fichier texte et envoyez-le à votre voisin chiffré avec sa clef publique. Lui vous enverra un fichier chiffré avec sa clef. Renvoyez-lui le message qu'il vous a envoyé mais en clair.

3ème partie : Création des certificats

Créez un répertoire appelé ssl. Vous vous placerez dans ce répertoire afin que les clés et les certificats soient créés à l'intérieur avant d'effectuer les manipulations.

Création du certificat serveur

Génération de la clé privée

On génère la clef privée avec la commande suivante en définissant un nom de fichier :

```
openssl genrsa 1024 > servwiki.key
```

La sortie attendue est la suivante :

```
Generating RSA private key, 1024 bit long modulus
 ..................++++++
 ..........................................................++++++
 e is 65537 (0x10001)
```

Si vous souhaitez que cette clé ait un mot de passe (qui vous sera demandé à chaque démarrage d'apache, donc à éviter !), ajoutez "-des3" après "genrsa".

Ceci a pour effet de créer une clé SSL (fichier servwiki.key), ne la perdez pas... **c'est votre clé privée**... (ni éventuellement le mot de passe) !

Vous pouvez observer son contenu : less servwiki.key

```
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQDQG9wvnuLC4aqzaJCAWGA1AxFzg00hjPObhq1mukzsGyuuWBFG
vj/k9vFNYX55DHctb/4cXtsZRWWvgcjtYnCVwRu+DAjFsk//kOMfhplmiv9xQ+ZL
8w/Xrnm8JWdSS+S4LCMnsuIiQtLbhMrQnUV02hAtbIZiSM3k6OjShEZhDQIDAQAB
AoGAHi0cBW+1k+qjFPbBlUq7UJSMUEKmyYmlvVSPCklTZB0gfVxZzPdDTpEcNks/
yo+rLFSD9Vsvy/9LGmLoXruadWlK67PCUnpM5/oRRGgy8t73YKrxflAU5Gtymjvc
ZCf0CAs6wBft3yLU31Qc4WqVM2vTyUH76jebVhxEw8k63OUCQQD/1OmAXV+TxBPG
ZTPFbzUeAE5rQqqOW4aoMNvM61Yn/19h6SzY2MfSQvF1BNns/efCRrqOMeyvPWUG
g1okfogTAkEA0D7pDf/D2Yu5msbOAGF4QBU1erLzpi/s6Rv6VEPYCGnHQlo3jbg9
FZbjHJ4UcYyYaA8jIrkY+FIJM88YlGbWXwJBAILEdvJ5R/CFCkKf2j2yIWmLaIol
En8fw43XI5L0PB7Hxx6KDLVu4XzVYQyahTZBdqR0eMlUNZJBhJE2tO3wi2cCQQCp
JkCFd3es0BrNxqfzlThozRFofcz88za7TldydL0YcFtC4Sb4vWsYizwktZ6jcPEm
rQz8Gl9W7MO+ynwLptB/AkEA1tsnFXoYzI71enmTdugGxbv0RqAd5iQpDYQkDSdn
2LImp/3YnXNJ9qpY91j87tKthh/Oetu6SHlmLg1LOYNIdw==
-----END RSA PRIVATE KEY-----
```

Protégez votre fichier en faisant : chmod 400 servwiki.key

De multiples autres options sont possibles mais il est inutile d'alourdir le sujet (à creuser pour une éventuelle PTI...)

A partir de votre clé, vous allez maintenant créer un fichier de demande de signature de certificat (CSR Certificate Signing Request).

On génère la demande de certificat avec la commande suivante :

```
openssl req -new -key servwiki.key > servwiki.csr
```

Le système va vous demander de saisir des champs ; remplissez-les en adaptant sauf le champ "Common Name" qui doit être identique au nom d'hôte de votre serveur virtuel :

```
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]: NORMANDIE
Locality Name (eg, city) []:ROUEN
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ESIGELEC
Organizational Unit Name (eg, section) []:TIC
Common Name (eg, YOUR name) []:wiki.domain1.org
Email Address []:
```

Ce n'est pas la peine de saisir d'autres "extra attributes"...

Ceci a pour effet de créer le formulaire de demande de certificat (fichier servwiki.csr) à partir de notre clé privée préalablement créée.

Ce fichier contient la clé publique à certifier. Un less servwiki.csr nous donne :

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBsTCCARoCAQAwcTELMAkGA1UEBhMCRlIxFTATBgNVBAgTDENvcnNlIGR1IFN1
ZDEQMA4GA1UEBxMHQWphY2NpbzEMMAoGA1UEChMDTExCMREwDwYDVQQLEwhCVFMg
SU5GTzEYMBYGA1UEAxMPcHJvZi5idHNpbmZvLmZyMIGfMA0GCSqGSIb3DQEBAQUA
A4GNADCBiQKBgQDSUagxPSv3LtgDV5sygt12kSbN/NWP0QUiPlksOkF2NkPfwW/m
f55dD1hSndlOM/5kLbSBo5ieE3TgikF0IktjBWm5xSqewM5QDYzXFt031DrPX63F
vo+tCKTQoVItdEuJPMahVsXnDyYHeUURRWLWwc0BzEgFZGGw7wiMF6wt5QIDAQAB
oAAwDQYJKoZIhvcNAQEEBQADgYEAwwI4UvkfhBvyRrUXtjrLfZXLxZlF9o+URmHJ
ysvrLKfesVBEzdA9mqk1OwIwLfe8Fw2fhip2LGqvcCPxDMoIE/0cDkqGRg9iKp7D
DMuy69lPTEB6UtpVKO/1eage0oug6VqdfGAYMMSGyWFuO9FE4UE6HspVodb20wGV
4H8qZuk=
-----END CERTIFICATE REQUEST-----
```

Maintenant, deux choix s'offrent à vous :

- envoyer le fichier servwiki.csr à un organisme (le tiers de confiance ou **l'autorité de certification (CA)**) et ainsi obtenir le certificat dûment signé par la clé privée de l'organisme (après avoir payé),
- **ou bien signer vous-même le certificat.**

Ce dernier choix a notre préférence...

Création du certificat de l'autorité de certification

Pour signer un certificat, vous devez devenir votre propre autorité de certification, cela implique donc de réaliser une clé et un certificat auto-signé.

La création de la clé privée de l'autorité de certification se fait comme précédemment :

```
openssl genrsa -des3 1024 > ca.key
```

Ce qui a pour effet de créer la clé privée de l'autorité de certification. Dans ce cas, il vaut mieux rajouter l'option -des3 qui introduit l'usage d'une "passphrase" (mot de passe long qui peut même contenir du blanc) car c'est cette clé privée qui signera tous les certificats que l'on emettra ; cette "passphrase" sera donc demandée à chaque utilisation de la clé.

Ensuite, à partir de la clé privée, on crée un certificat x509 pour une durée de validité d'un an auto-signé :

```
openssl req -new -x509 -days 365 -key ca.key > ca.crt
```

Il faut saisir la passphrase... puisqu'on utilise "ca.key" que l'on a protégé. Et on donne les renseignements concernant cette fois-ci l'autorité de certification (c'est à dire nous-même).

Attention : le nom de "Common Name" doit être différent de celui qui a été donné pour la clé.

```
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:NORMANDIE
Locality Name (eg, city) []:ROUEN
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ESIGELEC
Organizational Unit Name (eg, section) []: TIC
Common Name (eg, YOUR name) []:cert_CA
Email Address []:
```

C'est notre certificat d'autorité de certification qui va permettre de signer les certificats créés.

La signature du certificat serveur par le CA (Certificate Autority)

La commande qui signe la demande de certificat est la suivante :

```
openssl x509 -req -in servwiki.csr -out servwiki.crt -CA ca.crt -CAkey
ca.key\
-CAcreateserial -CAserial ca.srl
```

L'option CAcreateserial n'est nécessaire que la première fois.

Il faut saisir la passphrase...

Le certificat signé est le fichier "servwiki.crt". La sortie de la commande attendue est la suivante :

```
Signature ok
subject=/C=FR/ST=NORMANDIE/L=ROUEN/O=ESIGELEC/OU=TIC/CN=wiki.domain1.org
Getting CA Private Key
Enter pass phrase for ca.key:
```

Vous avez maintenant un certificat pour votre serveur se nommant servwiki.crt.

```
less servwiki.crt
-----BEGIN CERTIFICATE-----
MIICVDCCAb0CAQEwDQYJKoZIhvcNAQEEBQAwdDELMAkGA1UEBhMCRlIxFTATBgNV
BAgTDENvcnNlIGR1IFN1ZDEQMA4GA1UEBxMHQWphY2NpbzEMMAoGA1UEChMDTExC
```

```
MREwDwYDVQQLEwhCVFMgSU5GTzEbMBkGA1UEAxMSc2VydmV1ci5idHNpbmZvLmZy
MB4XDTA0MDIwODE2MjQyNloXDTA0MDMwOTE2MjQyNlowcTELMAkGA1UEBhMCRlIx
FTATBgNVBAgTDENvcnNlIGR1IFN1ZDEQMA4GA1UEBxMHQWphY2NpbzEMMAoGA1UE
ChMDTExCMREwDwYDVQQLEwhCVFMgSU5GTzEYMBYGA1UEAxMPcHJvZi5idHNpbmZv
LmZyMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDSUagxPSv3LtgDV5sygt12
kSbN/NWP0QUiPlksOkF2NkPfwW/mf55dD1hSndlOM/5kLbSBo5ieE3TgikF0Iktj
BWm5xSqewM5QDYzXFt031DrPX63Fvo+tCKTQoVItdEuJPMahVsXnDyYHeUURRWLW
wc0BzEgFZGGw7wiMF6wt5QIDAQABMA0GCSqGSIb3DQEBBAUAA4GBALD640iwKPMf
pqdYtfvmLnA7CiEuao60i/pzVJE2LIXXXbwYjNAM+7Lov+dFT+b5FcOUGqLymSG3
kSK6OOauBHItgiGI7C87u4EJaHDvGIUxHxQQGsUM0SCIIVGK7Lwm+8e9I2X0G2GP
9t/rrbdGzXXOCl3up99naL5XAzCIp6r5
-----END CERTIFICATE-----
```

Il faut maintenant **installer le certificat de l'autorité de certification dans chaque navigateur client**. C'est ce dernier qui va valider le certificat reçu par le client lors de la requête https://.

Installation du certificat d'autorité de certification

Taper la commande:

```
openssl pkcs12 -export -out server.p12 -inkey server.key -in server.crt
-certfile ca.crt
```

Pour installer sur votre navigateur le certificat de l'autorité de certification, fichier ca.crt :

- "Arrangez-vous" pour avoir le certificat disponible à partir du client (disquette, clé usb, ftp, ssh, répertoire partagé...)
- Sous Windows, un clic droit ou double-clic sur le fichier devrait vous permettre de lancer l'assistant d'installation du certificat dans Internet Explorer. **Vous devez l'installer en tant qu'autorité de certification**. Vérifiez ensuite que le certificat s'y trouve bien sous le nom de "cert_CA".
- sur Mozilla :
    - Edition/préférence/Confidentialité et Sécurité/Certificats
    - Bouton de commande : gestion des certificats
    - Onglet : autorité
    - Bouton de commande : importer
    - etc...

Il ne reste plus maintenant qu'à configurer apache2.

Etape 2 : configuration d'Apache2

Activation du module ssl

Il est nécessaire dans un premier temps d'activer le module ssl :

```
a2enmod ssl
Module ssl installed; run /etc/init.d/apache2 force-reload to enable
```

Comme on vous le demande, relisez la configuration du serveur.

La commande ls /etc/apache2/mods-enabled/ donne partiellement ceci :

```
ssl.conf
ssl.load
```

Configuration du port

Ajoutez au fichier /etc/apache2/ports.conf la ligne suivante :

```
Listen 443
```

Configuration du virtual host

Nous allons faire fonctionner sur notre serveur simultanément des virtualhost fonctionnant sous HTTP (sur le port 80) et des virtualhost fonctionnant sous HTTPS(sur le port 443). *Dès lors que l'on a des serveurs virtuels qui "tournent" sur des ports différents, il est obligatoire de préciser pour chaque serveur virtuel son port* . Aussi, dans le fichier `/etc/apache2/site-available/default` , il faut modifier :

```
NameVirtualHost * devient  NameVirtualHost *:80
<VirtualHost *> devient <VirtualHost *:80> pour chaque virtualhost
<VirtualHost adr_IP> devient <VirtualHost adr_IP:80> pour chaque
virtualhost
```

Profitez-en pour effacer le virtualhost correspondant au ServerName : wiki.domain1.org puisque c'est celui que l'on va configurer en HTTPS.

Il est possible de rajouter la configuration du virtualhost devant fonctionner sous ssl dans le fichier `/etc/apache2/site-available/default`, mais nous allons garder l'esprit modulaire et créer un fichier de conf spécifique `/etc/apache2/site-available/default-ssl`.

Voici ce que vous devez trouvez au minimum dans le fichier `/etc/apache2/site-available/default-ssl` (les commentaires et/ou explications éventuelles précèdent les directives)

```
#La déclaration du virtualhost.
#Attention : mettre bien une "*" et NON une adresse IP car
#on ne peut trouver qu'une seule fois la directive
#NameVirtualHost adresseIP :Port et nous l'avons déjà dans
#le fichier "default".

NameVirtualHost *:443

#Du coup : mettre aussi ici une "*" car sinon vous allez
#avoir le warning suivant très horripilant :
#NameVirtualHost *:443 has no VirtualHosts

<VirtualHost *:443>
ServerName wiki.domain1.org
DocumentRoot /home/web/wiki
```

```
#Les directives pour les certificats

SSLEngine On
SSLCertificateFile    /etc/apache2/ssl/servwiki.crt
SSLCertificateKeyFile /etc/apache2/ssl/servwiki.key

#Les directives qui suivent ne sont pas obligatoires mais
#aident au débugage
#On isole les erreurs relatives à http
#Attention : il faut créer le fichier "error_ssl.log"

ErrorLog /var/log/apache2/error_ssl.log
LogLevel warn

</VirtualHost>
```

Il est nécessaire maintenant d'activer le site : **a2ensite default-ssl**

Attention : ne pas oublier de relire ensuite la configuration de même qu'à chaque modification ultérieure du fichier.

Vous pouvez maintenant "passer" à la phase de test...

A partir d'un navigateur sur votre poste client :

```
https://wiki.domain1.org
```

Votre page devrait s'afficher normalement.

Si vous avez une alerte de sécurité, c'est que vous avez mal réalisé une étape. Selon l'alerte, reprenez l'étape correspondante.

Si ça ne fonctionne pas vous pouvez même mettre le LogLevel à "debug" ou "info" à la place de "warn" puis dans une console : "tail -f /var/log/apache2/error_ssl.log" pour voir quels sont les problèmes qui surviennent quand vous rechargez la lecture de la configuration.