

Travaux Pratiques R&T 2^{ème} année
Durée : 6 heures
TP R2b - SECURITE RESEAUX
SSL - Certificats - Signature - Apache - HTTPS

Noms : CARRETTE BAUD
Groupe : TP3
Date : 25/11/11

Objectifs du TP

Ce TP est composé de deux parties, la première a pour objectifs :

- Découvrir le protocole SSL/TLS et openssl
- Création de certificats, chiffrement et signature à l'aide de openssl.

Deuxième partie a pour objectifs :

- Installation et configuration d'un serveur Web Apache sous Linux.
- Configuration d'apache pour utiliser SSL

Vous en utiliserez pour ce TP **un PC sous Linux** (installation du serveur Apache Http et Https) et **un sous Windows/Linux** (rédaction du compte-rendu et client de test).

Vous détaillerez toutes vos démarches et vous fournissez des captures d'écran des résultats des commandes.

1.1 Protocole SSL/TLS

Qu'est-ce que SSL (ou son nouveau nom TLS) ?

SSL (Secure Socket Layer) est un protocole qui sert à sécuriser les échanges via Internet, devenu en 2001 TLS (Transport Layer Security), mais ayant pour objectifs communs lors d'un échange client-serveur :

- confidentialité des données échangées (notamment en chiffrant les données)
- intégrité des données
- authentification du serveur, ainsi que du client via l'utilisation d'un certificat

Quel(s) algorithme(s) de chiffrement sont utilisés avec SSL ?

Lors de l'utilisation de SSL, on peut utiliser différents systèmes de chiffrement (qui correspondent chacun à des algorithmes plus ou moins importants) :

- un système de chiffrement asymétrique (comme RSA ou Diffie-Hellman). Il est utilisé pour générer la clé principale qui permettra de générer des clés de session, d'où l'appellation asymétrique.

–un système de chiffrement symétrique (DES, 3DES, IDEA, RC4...) en utilisant les clés de session pour chiffrer les données.

Le client et le serveur se mettent d'accord au début de leur échange sur le choix de l'algorithme.

Qu'est-ce donc que HTTPS ? Sur quel port « écoute » HTTPS ?

HTTPS est un protocole variant de HTTP, qui ajoute à celui-ci l'usage du protocole TLS (voir question précédente) afin de sécuriser les échanges web entre un client et un serveur grâce à l'utilisation d'un certificat d'authentification.

HTTPS écoute par défaut sur le port 443.

1.2 OpenSSL

Qu'est ce que openssl ?

OpenSSL est un utilitaire open source qui regroupe plusieurs outils afin de permettre la cryptographie via les protocoles SSL et TLS en ligne de commande.

Quelles sont ses principales fonctionnalités ?

- OpenSSL permet de réaliser les fonctions suivantes :
 - o Création de paramètres des clefs RSA, DH et DSA
 - o Création de certificats X.509, CSRs et CRLs
 - o Calcul de signature de messages
 - o Chiffrement et Déchiffrement
 - o Tests SSL/TLS client et server
 - o Gestion de mail S/MIME signé ou chiffrés

Installez les paquets suivants (si nécessaire selon les dépendances) :

```
–openssl  
–libssl  
–ssl-cert
```

Sous le PC Linux, on entre les commandes suivantes permettant l'installation des paquets souhaités :

```
apt-get install openssl  
apt-get install ssl-cert
```

A noter qu'il n'y a pas besoin du paquet « libssl ». En effet lors de l'installation de celui-ci, il nous est indiqué que celui-ci est devenu obsolète au fil du temps, et que par conséquent on peut s'en passer.

1.3 Certificats

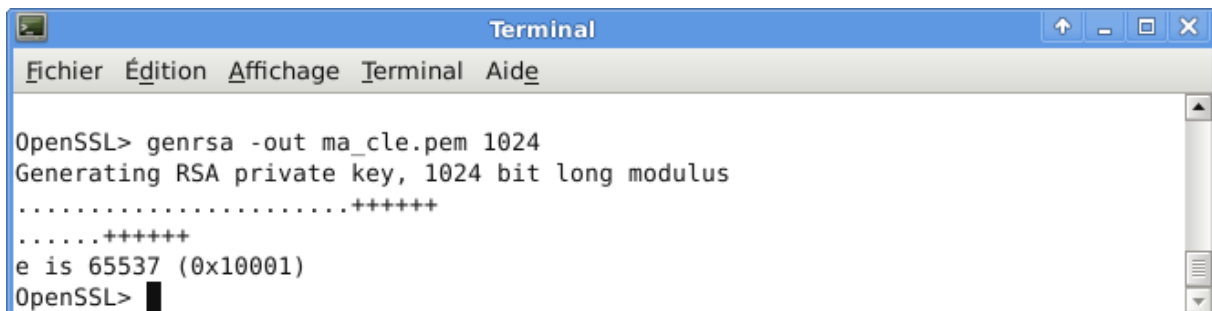
Qu'est ce qu'un certificat X.509 ?

Un certificat X.509 est une norme de cryptographie de l'Union internationale des télécommunications pour les infrastructures à clés publiques . Ce certificat établit les formats standard de certificats électroniques ainsi qu'un algorithme pour la validation de chemin de certification.

Citez quelques extensions de certificats ?

- .csr : Demande de certificat ;
- .crt : certificat public ;
- .key : clé privée du certificat ;
- .p12 : Certificat personnel contenant l'intégralité des éléments privés et publics ;

Avec la commande *genrsa* de openssl, générer une paire de clé (publique/privée) RSA de taille 1024 bits et stocker le résultat dans un fichier *ma_Cle.pem*.



```
Terminal
Fichier Édition Affichage Terminal Aide
OpenSSL> genrsa -out ma_cle.pem 1024
Generating RSA private key, 1024 bit long modulus
.....++++++
.....++++++
e is 65537 (0x10001)
OpenSSL> █
```

- genrsa* : création de la clé rsa
- out* : option qui définit dans quel fichier sera stockée la clé rsa.
- ma_cle.pem* : fichier dans lequel se trouve la clé.
- 1024* : Option qui définit la taille de la clé RSA en bits.

Visualisez (en utilisant les commande de openssl) le contenu du fichier *ma_Cle.pem* (considérée comme clé privée).

On affiche ici aussi le contenu de ce fichier en mode crypté, c'est-à-dire en effectuant la commande :
`cat ma_cle.pem`

The image shows a Linux desktop environment with a terminal window and a screenshot tool window.

Terminal Window (Left):

```

rc4          rc4-40

OpenSSL> rsa -in ma_cle.pem -text -noout
Private-Key: (1024 bit)
modulus:
 00:ce:fd:cc:07:d9:eb:ad:fc:d3:35:35:06:84:67:
 b5:36:56:57:03:29:e7:ef:b0:a1:eb:3f:bb:5b:e1:
 0d:a9:33:6f:d7:1f:65:05:b3:3c:9a:5a:21:ec:01:
 d0:1e:e7:f3:8f:de:9f:b6:44:b0:6a:bd:a1:68:f5:
 2f:28:77:8e:a0:a0:a2:82:5b:00:a1:2e:2b:39:4b:
 75:0e:55:da:df:92:c9:02:16:20:1a:82:86:89:0a:
 c1:eb:25:0e:b7:dd:07:ef:5e:cd:7e:e9:3d:bb:81:
 50:85:3b:33:c6:73:58:b7:79:cc:fa:f9:8b:1f:63:
 db:9f:d6:26:48:9b:f8:c2:93
publicExponent: 65537 (0x10001)
privateExponent:
 00:bc:f1:c1:3f:dd:75:9c:01:70:0e:5e:93:cf:63:
 17:70:3d:49:fa:12:53:ac:60:f0:bc:dc:15:3b:9e:
 68:a8:ed:d2:06:6d:9a:fa:a4:0d:0a:dc:81:a0:7c:
 5b:2d:0b:10:32:60:c6:2b:4b:f2:39:cd:55:bc:88:
 2a:fe:e6:7b:a2:92:0e:25:60:87:a2:11:79:6f:d6:
 15:2c:43:7d:5f:3d:0b:0c:6f:f6:7f:a8:9a:91:2c:
 e5:f0:43:49:21:33:3f:1a:5e:4d:d9:44:1d:1b:fa:
 d6:5d:be:3f:6e:ed:d8:98:30:96:94:a2:f9:6f:13:
 d6:ee:48:04:9d:a3:5c:99:09
prime1:
 00:ea:1d:eb:21:d0:3e:92:be:85:20:9a:18:59:69:
 3a:9f:5c:2e:7d:79:b2:8b:f2:06:d9:ea:ae:46:e7:
 13:b6:88:89:d1:6d:71:ee:76:6b:96:69:12:0a:c2:
 ff:45:c8:ff:46:1b:a2:a7:c8:b6:1a:36:72:cb:84:
 a6:ee:b0:e6:87
prime2:
 00:e2:56:ce:03:35:c1:e5:dd:70:bb:6c:2c:3f:bb:
 a8:fd:1e:e9:54:ac:7b:fc:4e:09:c3:e9:33:97:f6:
 d8:be:db:30:6a:20:31:53:fb:a9:0c:00:c0:3c:37:
 65:72:43:5f:60:3b:8b:31:88:22:bb:dd:1d:43:95:
 bb:78:f8:3a:95
exponent1:
 00:9b:3e:8f:a6:f2:72:2b:d8:ce:6a:9a:04:3f:75:
 f9:fb:a6:a2:e5:6d:87:aa:29:29:c5:e3:4b:01:95:
 45:8b:2a:eb:48:c6:ac:60:16:82:ad:50:ab:38:1b:
 aa:2e:da:63:fc:57:62:fa:a3:ab:ad:a2:21:69:37:
 63:82:b0:27:8d
exponent2:
 76:42:28:8d:a0:ce:d8:8d:64:d3:68:67:8e:3d:3c:
 c4:54:1c:51:b3:75:6c:94:51:a1:dc:fc:aa:0f:bd:
 9b:aa:e6:96:ec:8f:19:74:4a:15:0d:67:63:8f:06:
 20:37:77:63:ce:78:a1:be:a1:77:4a:c6:79:83:88:
 5a:62:48:29
coefficient:
 09:99:e8:be:89:8a:e6:25:1f:05:2d:19:18:09:96:
 f1:d8:2f:ea:30:8b:24:d7:f8:83:35:78:f3:bb:37:
 32:55:59:81:22:d2:53:a3:fb:c8:b5:37:ed:05:3c:
 bd:be:cb:37:7d:ac:62:86:fa:cf:c1:3e:a3:68:1d:
 14:7e:88:85

```

Screenshot Tool Window (Right):

The screenshot tool window is titled "A avec openssl" and shows a screenshot of the terminal output. The text visible in the screenshot is:

```

nt une paire de clés

la sortie normalement

```

Terminal Window (Bottom):

```

root@iutclrt:~# xfce4-screenshooter
(xfce4-screenshooter:3077): libxfce4util-CRITICAL **: xfce_rc_close: assertion `
rc != NULL' failed
root@iutclrt:~# cat ma_cle.pem
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQD0/cwH2eut/NM1NQaEZ7U2VlCdKefvsKHrP7tb4Q2pM2/XH2UF
szyaWiHsAdAe5/0P3p+2RLBqvaFo9S8od46goKKCWwChLis5S3U0vdrfkskCFiAa
goaJCsHrJQ633QfvXs1+6T27gVCF0zPGc1i3ecz6+YsfY9uf1iZIm/jCkwIDAQAB
AoGBALzxwT/ddZwBcA5ek89jF3A9SfoSU6xg8LzcFTueaKjt0gZtmvqkDQrcgaB8
Wy0LEDJgxitL8jnNVbyIKv7me6KSDiVgh6IRew/WFSxDfV89Cwxv9n+ompEs5fBD
SSEzPxpTdLEHRv61l2+P27t2JgwlPsi+W8T1u5IBJ2jXJkAJAEa6h3rIdA+kr6F
IJoYwWk6nlwufXmyi/IG2equRucTtoiJ0W1x7nZrLmkScsL/Rcj/Rhuip8i2GjZy
y4Sm7rDmhWJBA0JWzgM1weXdcLtsLD+7qP0e6VSse/x0CcPpM5f22L7bMGogMVP7
qQwAwDw3ZXJDX2A7izGIIrVdHU0Vu3j40pUCQQCbPo+m8nIr2M5qmgQ/dfn7pqlL
bYeqKSnF40sBLUWLKutIxqxFoKtUKs4G6ou2mP8V2L6o6utoiFpN20CsCeNAkB2
QiiNoM7YjWTTaGeOPTzEVbXrs3VsLFgh3PyqD72bquaw7I8ZdEoVdWdjYjYgN3dj
znhvqF3SsZ5g4haYkgpAKAJmei+iYrmJR8FLRkYCZbx2C/qMIsk1/iDNXjzuczcy
VVmBITJTo/vItTfBTy9vss3faxihvrPwT6jaB0Ufoif
-----END RSA PRIVATE KEY-----
root@iutclrt:~# verify
bash: verify : commande introuvable
root@iutclrt:~# xfce4-screenshooter

```

Pourquoi doit-on protéger ce fichier ?

Pour ne pas permettre à tout le monde de le lire. Sinon, il n'y a pas d'intérêt à crypter une information.

Protégez ce fichier en utilisant l'algorithme symétrique DES. Quel est l'autre moyen de protection qui existe ?

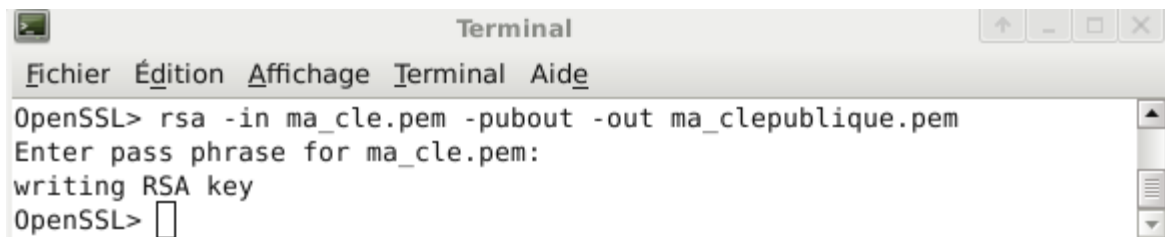
```
OpenSSL> rsa -in ma_cle.pem -des -out ma_cle.pem
writing RSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
OpenSSL> █
```

Rsa -in : cle rsa en entrée
ma_cle.pem : fichier source.
-des : définit l'algorithme de chiffrement voulu.
ma_cle.pem : fichier destination.
Phrase PEM choisie ici : « je suis un petit chaperon rouge » .

Idea est l'autre algorithme de chiffrement de la clé RSA, on peut donc l'utiliser en utilisant l'option -idea à la place de -des.

Extraire la partie publique de ma_Cle.pem et nommez la ma_ClePublique.pem. (A l'aide de l'option *pubout*).

rsa -in ma_cle.pem -pubout -out ma_clepublique.pem
in ma_cle.pem : fichier source
-pubout : option d'extraction de la partie publique.
-out ma_clepublique.pem : fichier de destination.



```
Terminal
Fichier Édition Affichage Terminal Aide
OpenSSL> rsa -in ma_cle.pem -pubout -out ma_clepublique.pem
Enter pass phrase for ma_cle.pem:
writing RSA key
OpenSSL> █
```

Créez une requête pour une demande de certification contenant votre clé publique (ma_demande.pem) (Mettez le champ common Name : www.test.com, [important pour le prochain tp!!](#))

req -new -key ma_cle.pem -in ma_clepublique.pem -out ma_demande.pem

En absence d'un organisme d'autorité de certification CA, c'est à vous de jouer son rôle :

-Générez une paire de clé (privée et publique) pour CA. (CA_cle.pem, CA_clepublique.pem).

On refait les mêmes étapes que pour la création d'une paire de clés pour le serveur (questions précédentes) :

```
genrsa -out CA_cle.pem 1024
```

```
root@iutclrt:~# openssl genrsa -out CA_cle.pem 1024  
Generating RSA private key, 1024 bit long modulus
```

```
.....+++++
```

```
.....+++++
```

```
e is 65537 (0x10001)
```

```
root@iutclrt:~# openssl rsa -in CA_cle.pem -pubout -out CA_clepublique.pem  
writing RSA key _
```

```
.....  
root@iutclrt:~# openssl rsa -in CA_cle.pem -des -out CA_cle.pem  
writing RSA key  
Enter PEM pass phrase:  
Verifying - Enter PEM pass phrase:
```

-Créez un certificat d'autorité de type X.509 pour CA avec un *common name* CA. (CA_certificat.pem)

```
req -new -x509 -days 365 -key CA_cle.pem > CA_certificat.pem
```

```
root@iutclrt:~# openssl req -new -x509 -days 365 -key CA_cle.pem > CA_certificat.pem  
Enter pass phrase for CA_cle.pem:  
You are about to be asked to enter information that will be incorporated  
into your certificate request.  
What you are about to enter is what is called a Distinguished Name or a DN.  
There are quite a few fields but you can leave some blank  
For some fields there will be a default value,  
If you enter '.', the field will be left blank.  
-----  
Country Name (2 letter code) [AU]:fr  
State or Province Name (full name) [Some-State]:  
Locality Name (eg, city) []:  
Organization Name (eg, company) [Internet Widgits Pty Ltd]:  
Organizational Unit Name (eg, section) []:  
Common Name (eg, YOUR name) []:CA  
Email Address []:
```

- new : indique que cette requête est nouvelle
- x509 : type de certificat
- key : clé à utiliser pour l'encryption

Créez votre certificat de type X.509 (mon_Certificat.pem) à partir de la demande de certification (ma_demande.pem). Votre certificat doit être signé par CA.

```
openssl x509 -req -in ma_demande.pem -out mon_certificat.pem -CA CA_certificat.pem -CAkey CA_cle.pem -CAcreateserial
```

```
root@iutclrt:~# openssl x509 -req -in ma_demande.pem -out mon_certificat.pem -CA CA_certificat.pem -CAkey CA_cle.pem -CAcreateserial
Signature ok
subject=/C=fr/ST=Some-State/O=Internet Widgits Pty Ltd/CN=www.test.com
Getting CA Private Key
Enter pass phrase for CA_cle.pem:
root@iutclrt:~#
```

On a donc créé notre certificat en utilisant la commande qui précède. On spécifie dans les options que celui-ci doit être signé par CA :

- CA : indique le certificat qui va vérifier
- CAkey : clé du certificat en question

Vérifiez la bonne signature de votre certificat.

(Commande : `openssl verify -CAfile CA_certificat.pem mon_Certificat.pem`)

```
root@iutclrt:~# openssl verify -CAfile CA_certificat.pem mon_certificat.pem
mon_certificat.pem: OK
root@iutclrt:~#
```

On constate à partir de ce résultat que le certificat a bien été signé.

1.4 Chiffrement/déchiffrement avec RSA

Créez un fichier `text_clair.txt` contenant des données à chiffrer. (ne doit pas dépasser 116 octets pour une clé de 1024 bits).

On crée un fichier que l'on nomme `text_clair.txt`, contenant à l'intérieur une phrase choisie aléatoirement, et qui sera chiffré par la suite.

Chiffrez le texte clair à l'aide de votre clé publique et mettez le résultat dans un fichier `text_chiffre.txt`.

Pour chiffrer le texte contenu dans le fichier énoncé dans la question précédente, on effectue la commande qui suit :

```
openssl enc -aes-256-cbc -in text_clair.txt -out text_chiffre.txt
```

Le résultat sera donc renvoyé dans un autre fichier, appelé `text_chiffre.txt`.

Déchiffrez le fichier `text_chiffre.txt` et mettez le résultat dans un fichier `text_dechiffre.txt`.

On déchiffre à présent notre fichier en effectuant cette commande :

```
openssl enc -aes-256-cbc -d -in text_chiffre.txt -out text_dechiffre.txt
```

Comparez les deux fichiers test_clair.txt et text_dechiffre.txt

Après comparaison, on constate que les deux fichiers sont bel et bien identiques, on peut donc en déduire que le chiffrement ainsi que le déchiffrement ont été corrects.

1.5 Signature

Qu'est ce que une fonction de hachage ?

Une fonction de hachage est une fonction qui comme son nom l'indique, va hacher les données pour en créer un condensé (dont la taille varie en fonction du type de la fonction utilisée) qui sera beaucoup moins important en taille.

Qu'est ce qu'une bonne fonction de hachage ? Donnez des exemples de bonnes fonctions de hachage.

Une bonne fonction de hachage crée des hach de taille différente en fonction de la taille du fichier en question.

Il n'est possible de signer que de petits documents. Pour signer un gros document on calcule d'abord une empreinte (à l'aide d'une fonction de hachage, utiliser la commande `dgst` pour le faire) de ce document. Signez un document revient à signer son empreinte.

Signez le sujet du TP puis vérifiez la signature.

```
root@iutclrt:~# openssl dgst -md5 -verify ma_clepublique.pem -signature signature.md5 SS
L-openssl-Appache.odt
Verified OK
```

1.6 Signature de courriers électroniques

But : Envoyer un courrier *signé* dont le contenu est votre sujet de TP à l'adresse mail de votre professeur.

Voir la commande `smime` d'`openssl` :

Entrée : le document à signer, votre certificat et votre clé privée.

Sortie : mail_signé.msg.

Pour vérifier un courrier signé, il faut disposer du certificat que l'émetteur a utilisé pour signer, ainsi que celui de l'autorité ayant émis ce certificat.

Envoyer à l'adresse mail de votre professeur :

–votre certificat

–le certificat de l'autorité de certification (CA)

–mail_signé.msg

- TRES IMPORTANT !!!!!

Sauvegarder vos fichiers (clés, certificats, ...) sur une clé USB. Vous allez en avoir besoin la prochaine fois pour continuer la deuxième partie.

Fin de la première partie

2.1 Installation et configuration classique de Apache

Installez les packages nécessaires pour apache sur la machine Linux.
`apt-get install apache2`

Configurez le champ `ServerName` avec la valeur www.test.com.

Dans quel fichier faites-vous la configuration d'apache ?

Il faut modifier le fichier `apache2.conf`. Le fichier `httpd.conf` également

Que faire pour que vos modifications soient prises en compte par le service Apache ?

Il faut ajouter une ligne au fichier `apache2.conf` : `ServerName` www.test.com

Puis ajouter dans le fichier `/apache2/site/available/default` la ligne :

`ServerName` www.test.com

Redémarrer le serveur apache

`/etc/init.d/apache2 restart`

Créez une page sur le serveur Web (dans `DocumentRoot`) appelée `test.html`.

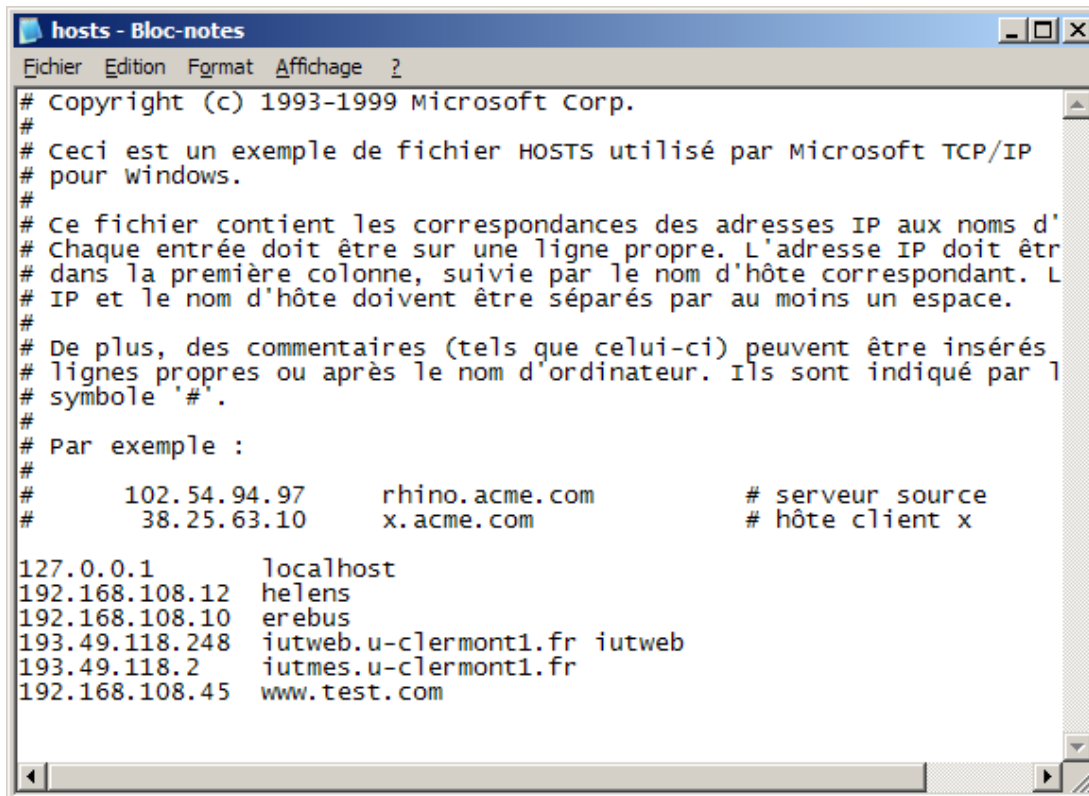


Depuis le client, faites un test pour vérifier le bon fonctionnement de votre page et de votre serveur apache. Comment vous faites ?

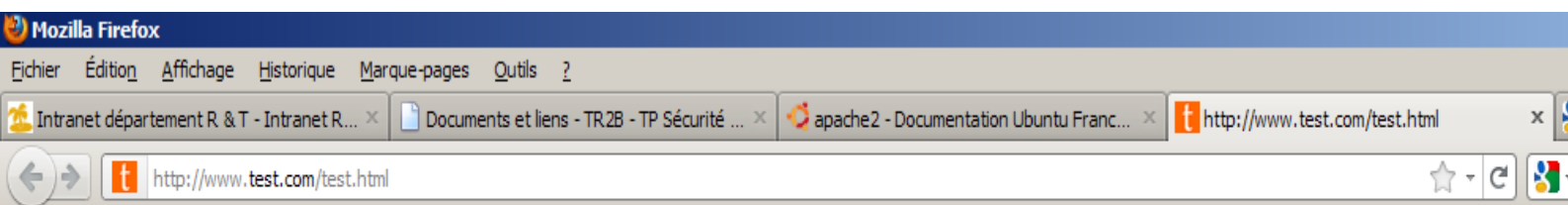
On retire le proxy, puis on se connecte au pc sous linux par le biais de son adresse IP qui est 192.168.108.49/test.html

Est-ce que www.test.com/test.html marche ? Que faut-il faire pour que ça marche ?

Non, cela ne fonctionne pas. Sur le pc client (sous windows on ajoute dans le fichier windows/system32/drivers/etc/hosts, et rajouter la ligne 192.168.108.45 www.test.com



```
hosts - Bloc-notes
Fichier Edition Format Affichage ?
# Copyright (c) 1993-1999 Microsoft Corp.
#
# Ceci est un exemple de fichier HOSTS utilisé par Microsoft TCP/IP
# pour Windows.
#
# Ce fichier contient les correspondances des adresses IP aux noms d'
# Chaque entrée doit être sur une ligne propre. L'adresse IP doit être
# dans la première colonne, suivie par le nom d'hôte correspondant. L'
# IP et le nom d'hôte doivent être séparés par au moins un espace.
#
# De plus, des commentaires (tels que celui-ci) peuvent être insérés
# lignes propres ou après le nom d'ordinateur. Ils sont indiqués par l
# symbole '#'.
#
# Par exemple :
#
#      102.54.94.97      rhino.acme.com      # serveur source
#      38.25.63.10     x.acme.com         # hôte client x
#
127.0.0.1      localhost
192.168.108.12 helens
192.168.108.10 erebus
193.49.118.248 iutweb.u-clermont1.fr iutweb
193.49.118.2   iutmes.u-clermont1.fr
192.168.108.45 www.test.com
```



(attention à la configuration du proxy dans le navigateur client).

2.2 Test de la sécurité de HTTP

Capturing from Broadcom NetXtreme Gigabit Ethernet Driver (Microsoft's Packet Scheduler) - Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: ip.addr == 192.168.108.46

No.	Time	Source	Destination	Protocol	Info
19	4.678019	192.168.108.46	192.168.108.45	TCP	mini-sql > http [SYN] Seq=0 win=65535 Len=0 MSS=1460 SACK_PERM=1
20	4.678117	192.168.108.45	192.168.108.46	TCP	http > mini-sql [SYN, ACK] Seq=0 Ack=1 win=5840 Len=0 MSS=1460 SACK_PERM=1
21	4.678134	192.168.108.46	192.168.108.45	TCP	mini-sql > http [ACK] Seq=1 Ack=1 win=65535 Len=0
22	4.678254	192.168.108.46	192.168.108.45	HTTP	GET /test.html HTTP/1.1
23	4.678404	192.168.108.45	192.168.108.46	TCP	http > mini-sql [ACK] Seq=1 Ack=477 win=6432 Len=0
24	4.679010	192.168.108.45	192.168.108.46	HTTP	HTTP/1.1 200 OK (text/html)
25	4.825305	192.168.108.46	192.168.108.45	TCP	mini-sql > http [ACK] Seq=477 Ack=380 win=65156 Len=0
46	19.668116	192.168.108.46	192.168.108.45	TCP	mini-sql > http [FIN, ACK] Seq=477 Ack=380 win=65156 Len=0
47	19.668336	192.168.108.45	192.168.108.46	TCP	http > mini-sql [FIN, ACK] Seq=380 Ack=478 win=6432 Len=0
48	19.668358	192.168.108.46	192.168.108.45	TCP	mini-sql > http [ACK] Seq=478 Ack=381 win=65156 Len=0

Frame 24: 433 bytes on wire (3464 bits), 433 bytes captured (3464 bits)

- Ethernet II, Src: Dell_85:24:f7 (00:26:b9:85:24:f7), Dst: Dell_85:eb:89 (00:26:b9:85:eb:89)
- Internet Protocol, Src: 192.168.108.45 (192.168.108.45), Dst: 192.168.108.46 (192.168.108.46)
- Transmission Control Protocol, Src Port: http (80), Dst Port: mini-sql (1114), Seq: 1, Ack: 477, Len: 379
- Hypertext Transfer Protocol
 - HTTP/1.1 200 OK\r\n
 - [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
 - [Message: HTTP/1.1 200 OK\r\n]
 - [Severity level: Chat]
 - [Group: Sequence]
 - Request Version: HTTP/1.1
 - Response Code: 200
 - Date: wed, 30 Nov 2011 08:58:16 GMT\r\n
 - Server: Apache/2.2.16 (Debian)\r\n
 - Last-Modified: wed, 30 Nov 2011 08:57:33 GMT\r\n
 - Etag: "2e598-25-4b2efed2e3940"\r\n
 - Accept-Ranges: bytes\r\n
 - Vary: Accept-Encoding\r\n
 - Content-Encoding: gzip\r\n
 - Content-Length: 42\r\n
 - Keep-Alive: timeout=15, max=100\r\n
 - Connection: Keep-Alive\r\n
 - Content-Type: text/html\r\n
 - \r\n
 - Content-encoded entity body (gzip): 42 bytes -> 37 bytes
 - Line-based text data: text/html
 - Blablabla !!\n
 - ouiiiiiiiiiiii\n
 - \n
 - Tralala\n

0000 00 26 b9 85 eb 89 00 26 b9 85 24 f7 08 00 45 00 .&....& ..\$.E.
 0010 01 a3 e4 45 40 00 04 06 fb 62 c0 a8 6c 2d c0 a8 ...E@.@. .b.]-..
 0020 6c 2e 0e 50 04 5a 0b 11 27 ac 82 66 54 c3 50 18].P.Z...'.ft.P.
 0030 10 20 d6 34 00 00 48 54 54 50 2f 21 2e 21 20 20 4 HT TP/1.1

Frame (433 bytes) Uncompressed entity body (37 bytes)

Frame (frame), 433 bytes Packets: 69 Displayed: 10 Marked: 0 Profile: Default

Avec Wireshark sur le PC client, scannez la même connexion à la page depuis le client. Pouvez-vous voir passer dans les trames le contenu de la page *test.html* créée précédemment ? Si oui, dans quel type de trames ?

Oui, on peut voir le contenu des trames en clair dans des trames HTTP

Qu'en déduisez-vous quand à la sécurité du protocole HTTP ? Pour quels types de données cela peut-il être critique ?

Si le texte apparaît en clair, c'est qu'il n'y a pas de sécurité. Cela peut être critique pour les données sensibles telles que les coordonnées bancaires...

2.3 Configuration de Apache

Il nous faut maintenant configurer apache pour qu'il utilise SSL.

Activez le module SSL de Apache2 (explorez pour cela la commande *a2enmod*). Vérifiez que cela a bien fonctionné en listant le contenu du répertoire */etc/apache2/mods-enabled*.

On utilise la commande *a2enmod*, et on lui met *ssl* lorsqu'il demande quel module activer sur apache.

Le fichier de configuration *ssl.conf* apparaît (répertoire *mods-enabled*):

```
root@none):/etc/apache2/mods-enabled# ls
alias.conf      authz_default.load  autoindex.conf    deflate.conf      env.load          negotiation.load  setenvif.load    status.load
alias.load      authz_groupfile.load autoindex.load    deflate.load      mime.conf        reqtimeout.conf  ssl.conf
auth_basic.load authz_host.load     cgid.conf         dir.conf         mime.load        reqtimeout.load  ssl.load
authn_file.load authz_user.load     cgid.load         dir.load         negotiation.conf setenvif.conf    status.conf
root@none):/etc/apache2/mods-enabled#
```

Configurez apache pour qu'il écoute en plus sur le port 443 (port du HTTPS par défaut).

Modifiez la configuration d'Apache pour que le répertoire *DocumentRoot* soit accessible en https (SSL). Explorez notamment les directives suivantes : *SSLEngine*, *SSLCertificateFile* et *SSLCertificateKeyFile*.

Configuration :

```
Terminal
Fichier Édition Affichage Terminal Aide
<VirtualHost 192.168.108.45:443>
  ServerAdmin webmaster@localhost
  ServerName www.test.com
  DocumentRoot /etc/apache2/
  SSLEngine on
  SSLCertificateFile /etc/ssl/mon_certificat.pem
  SSLCertificateKeyFile /etc/ssl/ma_cle.pem
  SSLCACertificatePath /etc/ssl
  SSLCACertificateFile /etc/ssl/CA_certificat.pem
</VirtualHost>

~
~

root@(none):/etc/apache2/sites-available# /etc/init.d/apache2 restart
Restarting web server: apache2Apache/2.2.16 mod_ssl/2.2.16 (Pass Phrase Dialog)
Some of your private key files are encrypted for security reasons.
In order to read them you have to provide the pass phrases.

Server www.test.com:443 (RSA)
Enter pass phrase:

OK: Pass Phrase Dialog successful.

root@(none):/etc/apache2/sites-available#
```

2.4 Installation du certificat d'autorité de certification sur le client

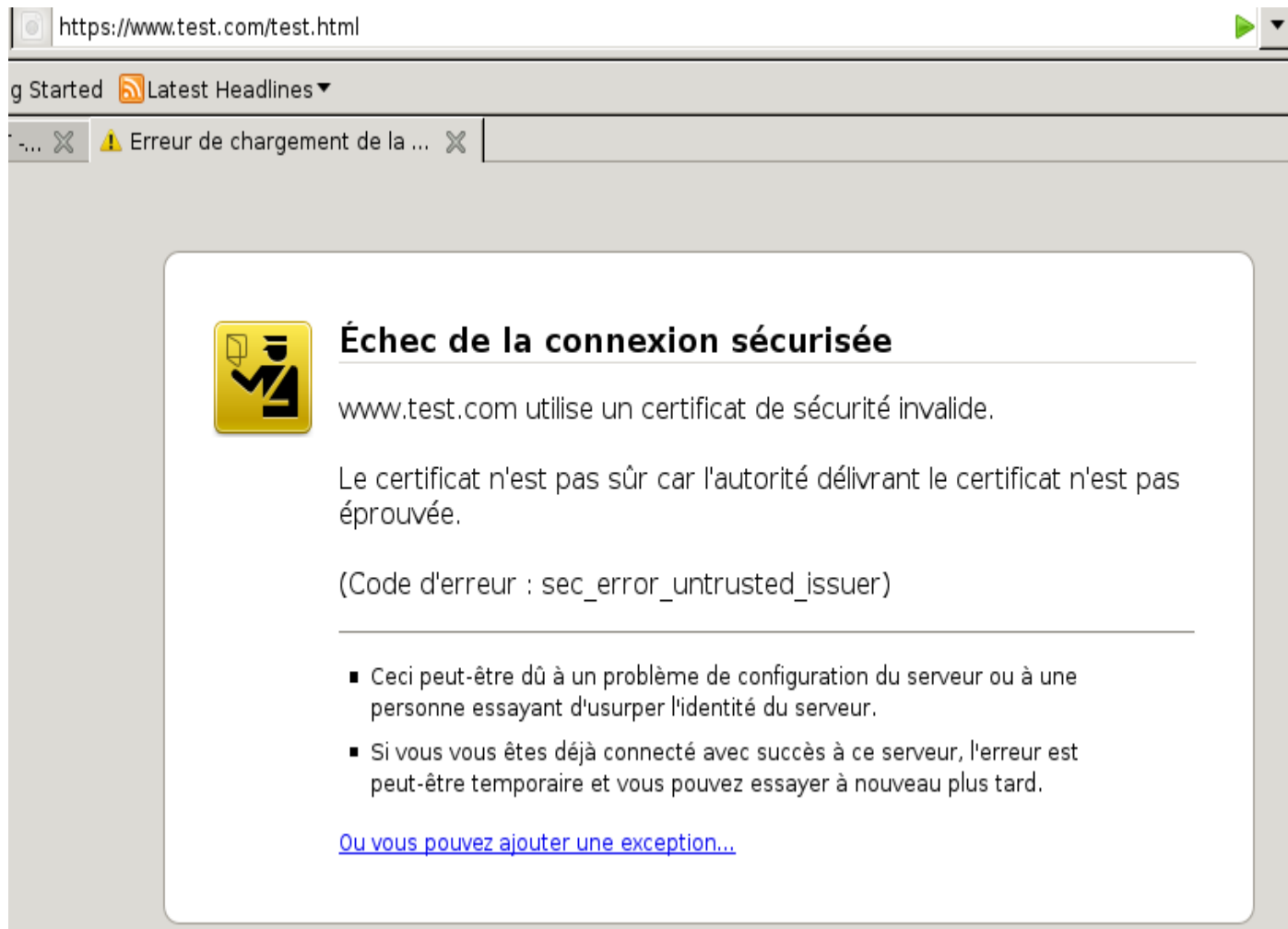
Le navigateur client ne connaît pas par défaut l'autorité de certification qui a signé le certificat de votre serveur (seules certaines autorités de certifications dites de confiance peuvent signer des certificats serveurs, ce qui permet de ne pas retrouver une alerte dans le navigateur client).

Installez donc dans votre navigateur le certificat de l'autorité de certification que vous avez créé auparavant.

Donnez votre démarche pour Firefox et Internet Explorer.

Démarche Firefox :

Au départ, la connexion au site web <https://www.test.com/test.html> est un échec car l'autorité de certification n'est pas reconnue comme valide par le navigateur. Il faut donc installer le certificat du CA.




The screenshot shows a web browser window with the address bar containing "https://www.test.com/test.html". The browser interface includes a search bar, a "Latest Headlines" button, and a tab titled "Erreur de chargement de la ...". The main content area displays a yellow warning icon of a person with a shield, followed by the heading "Échec de la connexion sécurisée". Below the heading, the text states: "www.test.com utilise un certificat de sécurité invalide. Le certificat n'est pas sûr car l'autorité délivrant le certificat n'est pas éprouvée." The error code "(Code d'erreur : sec_error_untrusted_issuer)" is shown. A list of two bullet points explains the cause: "Ceci peut-être dû à un problème de configuration du serveur ou à une personne essayant d'usurper l'identité du serveur." and "Si vous vous êtes déjà connecté avec succès à ce serveur, l'erreur est peut-être temporaire et vous pouvez essayer à nouveau plus tard." A blue link "Ou vous pouvez ajouter une exception..." is provided at the bottom.

https://www.test.com/test.html

g Started Latest Headlines ▾

... ✕ ⚠ Erreur de chargement de la ... ✕

 **Échec de la connexion sécurisée**

www.test.com utilise un certificat de sécurité invalide.

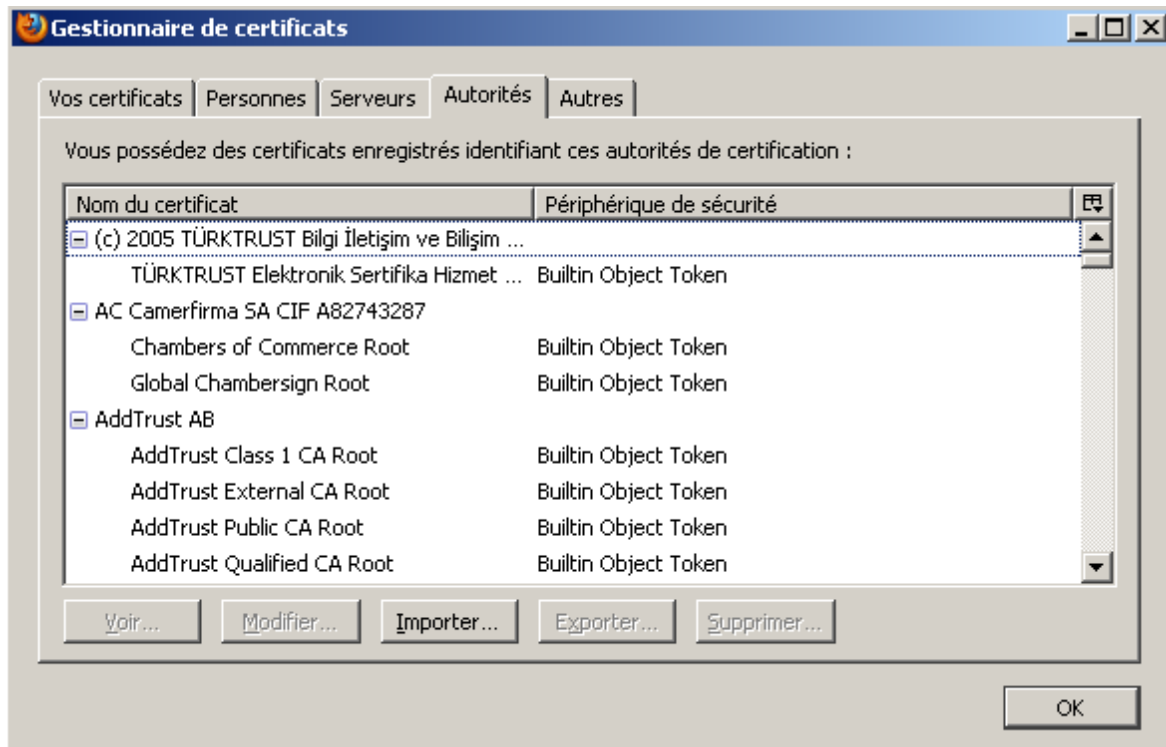
Le certificat n'est pas sûr car l'autorité délivrant le certificat n'est pas éprouvée.

(Code d'erreur : sec_error_untrusted_issuer)

- Ceci peut-être dû à un problème de configuration du serveur ou à une personne essayant d'usurper l'identité du serveur.
- Si vous vous êtes déjà connecté avec succès à ce serveur, l'erreur est peut-être temporaire et vous pouvez essayer à nouveau plus tard.

[Ou vous pouvez ajouter une exception...](#)

Afin d'installer notre certificat d'autorité nous suivons la procédure suivante :
On va dans Préférences → Avancées → Chiffrement → Voir liste des certificats
On importe notre certificat à la liste existante.



Le certificat est importé, la connexion peut se faire.

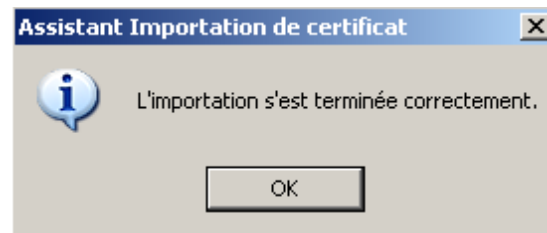
Démarche Internet Explorer :

Pour Internet Explorer, il faut suivre le chemin suivant :

Options → Options internet → Contenu → Certificat

Puis on fait Importer, on va dans Parcourir et on sélectionne notre certificat.

On obtient le message suivant confirmant la bonne importation du certificat.



2.5 Test de la sécurité de HTTPS

Depuis le client, lancez une connexion à l'adresse suivante :

<https://www.test.com/test.html>

Cela fonctionne-t-il correctement ?



Cela fonctionne, notre site est indiqué comme étant « sûr ».

En cas de problème, pensez-bien à regarder ce qui se passe dans les fichiers de log d'apache sur le serveur.

Avec Wireshark sur le PC client, scannez la même connexion que précédemment. Pouvez-vous voir passer dans les trames le contenu de la page *test.html* ? Qu'en déduisez-vous quand à la sécurité du protocole HTTPS ?

Avec wireshark, on remarque des échanges de trames ssl, ainsi que des demandes de certificat (Server Key Exchange). De plus, les données ne circulent plus en clair sur le réseau.

No.	Time	Source	Destination	Protocol	Info
37	15.630800	192.168.108.45	192.168.108.46	TCP	https > evm [ACK] Seq=1 Ack=166 win=6432 Len=0
38	15.638538	192.168.108.45	192.168.108.46	TLSv1	Server Hello, Certificate
39	15.638559	192.168.108.45	192.168.108.46	TLSv1	Server Key Exchange, Server Hello Done
40	15.638574	192.168.108.46	192.168.108.45	TCP	evm > https [ACK] Seq=166 Ack=1777 win=65535 Len=0
41	15.644990	192.168.108.46	192.168.108.45	TLSv1	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
42	15.650288	192.168.108.45	192.168.108.46	TLSv1	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
43	15.650722	192.168.108.46	192.168.108.45	TLSv1	Application Data
44	15.651401	192.168.108.45	192.168.108.46	TLSv1	Application Data, Application Data, Application Data, Application Data
45	15.662933	192.168.108.46	192.168.108.45	TLSv1	Application Data
46	15.663499	192.168.108.45	192.168.108.46	TLSv1	Application Data, Application Data, Application Data, Application Data
47	15.685443	192.168.108.46	192.168.108.45	TLSv1	Application Data
48	15.685947	192.168.108.45	192.168.108.46	TLSv1	Application Data, Application Data, Application Data, Application Data
49	15.889820	192.168.108.45	192.168.108.46	TLSv1	[TCP Retransmission] Application Data, Application Data, Application Data, A
50	15.889839	192.168.108.46	192.168.108.45	TCP	evm > https [ACK] Seq=1499 Ack=3815 win=65535 Len=0
51	15.928113	192.168.108.45	192.168.108.46	TCP	evm > https [ACK] Seq=1499 Ack=3815 win=65535 Len=0

Frame 39: 370 bytes on wire (2960 bits), 370 bytes captured (2960 bits)
 Ethernet II, Src: Dell_85:24:f7 (00:26:b9:85:24:f7), Dst: Dell_85:eb:89 (00:26:b9:85:eb:89)
 Internet Protocol, Src: 192.168.108.45 (192.168.108.45), Dst: 192.168.108.46 (192.168.108.46)
 Transmission Control Protocol, Src Port: https (443), Dst Port: evm (1139), Seq: 1461, Ack: 166, Len: 316
 [Reassembled TCP segments (411 bytes): #38(95), #39(316)]
 Secure Socket Layer

FIN DU TP (A FAIRE OBLIGATOIREMENT)

Supprimez tout ce que vous avez fait sur les deux PCs :

- Sur le PC Windows :

Désinstallez les certificats des autorités de certification dans les navigateurs.

- Sur le PC Linux :

Supprimez les certificats créés.

Supprimez les paquets installés :

apt-get remove --purge openssl

apt-get remove --purge libssl

apt-get remove --purge ssl-cert

apt-get remove --purge apache2-common

apt-get remove --purge apache2