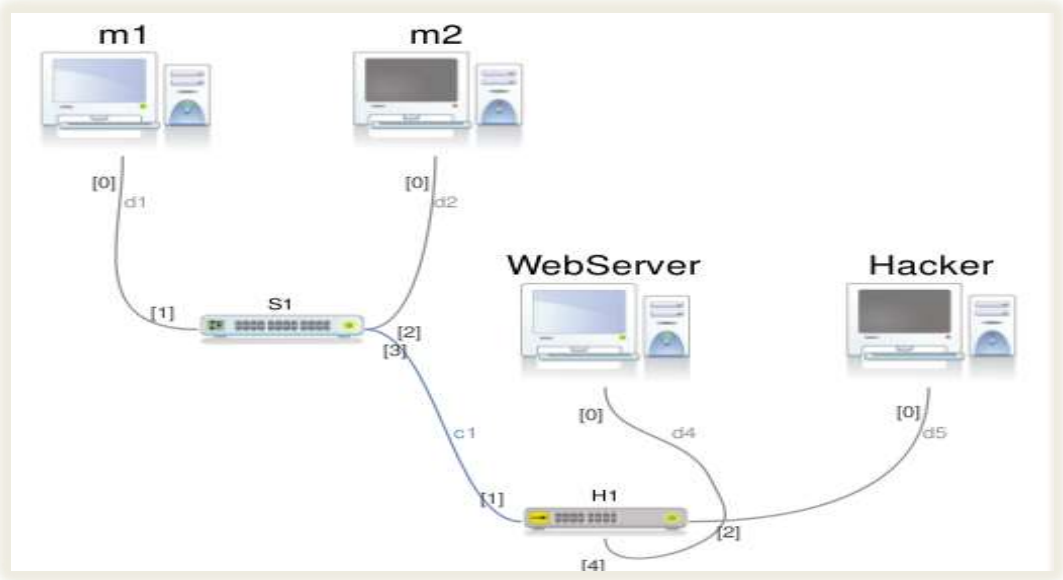
Université Paris 13 Travaux Pratiques
IUT de Villetaneuse RESEAUX
Département T & R

TP : Web/Apache/SSL

Fait par : *Noureddine AWANE*

1. Mise en place de la topologie réseau

Lancez Marionnet et créez la topologie réseau ci-dessous.



m1: 192.168.1.1/24
m2: 192.168.1.2/24
Webserver: 192.168.1.100/24
Hacker: 192.168.1.10/24

Avant de poursuivre le TP, assurez-vous que les différents composants L3 arrivent à se faire des Ping mutuellement.

2. Configuration du serveur Apache :

2.1. Installation du serveur

Sur la version Debian que vous utilisez Apache est déjà installé. Autrement pour installer Apache à titre d'information est :

```
> apt-get install apache2
ou
> aptitude install apache2
```

Quelle est la version de HTTP apache 2 qui a été installé sur votre système ?
Maintenant vérifiez si votre serveur http fonctionne. Vous devriez pourvoir faire afficher la première page web en allant à l'adresse ***http://192.168.1.100/***

Orange Restricted

2.2. Démarrage du serveur

Démarrer le serveur Apache avec la commande classique :

```
/etc/init.d/apache2 start
```

Pour les différents phases nécessaires (démarrage, arrêt, relance...etc.) pour la suite du TP, utilisez les scripts suivants :

```
> /etc/init.d/apache2 [start|restart|stop|reload|force-reload|status]
ou
> service apache2 [start|restart|stop|reload|force-reload|status]
```

2.3. configuration du serveur

Par défaut, les versions d'Apache n'utilisaient qu'un seul fichier de configuration nommé httpd.conf et situé dans le répertoire /etc/apache2/. Dans la version que nous avons (et toutes celles basées sur un système Debian), les informations du fichier httpd.conf sont réparties dans plusieurs fichiers.

Si le fichier httpd.conf est présent, vérifiez qu'il est bien vide à l'aide de la commande suivante :

```
> ls /etc/apache2/
> cat /etc/apache2/httpd.conf
```

De manière générale, il est conseillé de faire une copie de sauvegarde des fichiers avant toutes modifications.

Vérifiez si vos fichiers de configuration contiennent des erreurs de syntaxe avec la commande suivante :

```
> apache2ctl -t
```

Vérifiez les modules qui ont été compilés avec le serveur avec la commande suivante :

```
> apache2ctl -l
```

Rendez-vous dans le répertoire /etc/apache2/.

Ouvrez le fichier de configuration d'apache ***apache2.conf*** et parcourez-le.

Examinez votre propre fichier de configuration et essayez de comprendre par vous-même l'ensemble des directives présentes. Quand vous pensez avoir compris, essayez de répondre aux questions ci-dessous.

• Vérifiez si le numéro du processus linux du démon httpd (apache2 dans notre cas) contenu dans le fichier apache.pid correspond bien au premier processus apache2.

• Combien y a-t-il de processus apache2 actuellement fonctionnel?

• Que se passe-t-il sur les numéros des processus apache2 après l'exécution d'une commande de redémarrage ?

• Vérifiez dans le fichier le nom d'utilisateur et du groupe d'apache.

• Que contiennent les répertoires /etc/apache/mods-available, /etc/apache/mods-enabled, /etc/apache/ sites - available et /etc/apache/sites-enabled ?

3. Le protocole http

Cette partie a pour but de vous faire tester simplement les requêtes et réponses du protocole http. Utiliser le protocole ***telnet*** sur la machine cliente comme suite :

```
telnet 192.168.1.100 80
```

Orange Restricted

3.1. Première requête

Maintenant nous allons envoyer notre première requête http. Envoyez la commande suivante et vérifiez si vous obtenez bien la page web correspondante.

```
GET
```

Comme la requête n'est pas valide nous devrions avoir une erreur. Mais le serveur nous retourne normalement la page web sans rien d'autre. Ecrivons maintenant une requête http valide la plus simple possible. Pour rappel, une requête doit avoir la forme suivante (les informations en [] sont facultatives) :

```
<Méthode> <URI> HTTP/<Version>
[<Champ d'entête>: <Valeur>]
[<tab><Suite Valeur si >1024>]
Ligne_vide (CRLF)
```

La requête la plus simple est donc la suivante (n'oubliez pas la ligne vide):

```
GET / HTTP/1.0
```

Vérifiez que vous obtenez bien quelque chose comme cela pour un fichier xhtml:

```
HTTP/1.1 200 OK
Date: Tue, 20 Aug 2013 13:18:06 GMT
Server: Apache/2.2.22 (Debian)
Vary: Accept-Encoding
Content-Length: 51
Connection: close
Content-Type: application/xhtml+xml
<html><body><h1>It works! xhtml</h1></body></html>
```

Ou comme cela pour un fichier html :

```
HTTP/1.1 200 OK
Date: Tue, 20 Aug 2013 13:18:06 GMT
Server: Apache/2.2.22 (Debian)
Vary: Accept-Encoding
Content-Length: 51
Connection: close
Content-Type: text/html
<html><body><h1>It works!</h1></body></html>
```

3.2. Tests des codes d'erreur

Le but est de tester les différents codes d'erreurs du protocole HTTP.

Erreur 404

Commençons par le code 404 : page no found. Ce code est retourné quand la page demandée n'existe pas. Pour vérifier cela, envoyez la requête suivante en vérifiant que vous n'avez pas le fichier test à la racine de votre serveur

```
GET /test http/1.0
```

Vous deviez avoir une réponse d'erreur, analysez là.

Faites la même requête avec votre navigateur web, observez-vous une différence ?
http://192.168.1.100/test

Ecrivez les requêtes nécessaires pour observer les codes d'erreurs suivants.

- 200: OK
- 403: Forbidden
- 404: Not Found
- 400: Bad Request

Orange Restricted

4. Serveur Web multi-site : Virtual host

Le principe des Serveurs Virtuels consiste à faire fonctionner un ou plusieurs serveurs Web (comme www.iut-paris13.fr et www.univ-paris13.fr) sur une même machine.

4.1. Création des répertoires et pages

La première étape va consister à la création de deux répertoires respectifs pour chacun des 2 hôtes virtuels. Ces 2 répertoires devront être créé dans /var/www/ car ce répertoire constitue la base du serveur web.

```
WebServer:~# cd /var/www/
WebServer:/var/www# mkdir iut-paris13
WebServer:/var/www# mkdir univ-paris13
WebServer:/var/www# ls
index.html  iut-paris13  univ-paris13
```

Dans chacun de ces deux répertoires, créer une page d'accueil en php ou html.

4.2. Création des deux hôtes virtuels

Avant de poursuivre, regardons un peu ce que contient le fichier de configuration d'un hôte virtuel.
Ouvrez le fichier /etc/apache2/sites-available/default et examinez l'ensemble des paramètres de ce fichier.

Pour gagner du temps faire une copie du fichier «default» qui servira de base pour nos sites :

```
cp /etc/apache2/sites-avalaible/default /etc/apache2/sites-avalaible/iut-paris13
cp /etc/apache2/sites-avalaible/default /etc/apache2/sites-avalaible/univ-paris13
```

Quelques explications des sections qui nous intéressent dans la configuration

- `<VirtualHost *:80>` et `</VirtualHost>`: Signalent le début et la fin de la section du vhost, en écoute pour toutes les interfaces/adresses IP (*) sur le port **80**.
- `ServerAdmin webmaster@localhost` : adresse mail où envoyer les messages d'erreur. Devra donc être remplacée par la vôtre
- `ServerName`: Nom utilisé par le vhost, remplacez-le par le nom de votre site (ici **site-1**)
- `ServerAlias`: ServerAlias définie les autres sous domaines pour lesquels le vhost répondra.
- `DocumentRoot` : Répertoire de stockage du site (sa racine). Vous devrez donc modifier cette directive pour qu'elle pointe sur votre racine (ex: `/home/www/iut-paris13`)
- `DirectoryIndex index.php` : En l'absence de page spécifique demandée par l'internaute, c'est cette page que sera affichée, c'est donc en quelque sorte votre page d'accueil. A modifier donc si votre page d'accueil porte un nom différent (`default.html`, `accueil.php`, etc.)

Modifier le fichier configuration pour l'adapter à vos deux sites web.

4.3. Activation/désactivation des sites

Une fois l'adaptation des deux fichiers (iut-paris13 et univ-paris13) faite, il reste à les activer pour qu'Apache les prenne en compte. Pour faire utiliser l'outil ***a2ensite*** :

```
a2ensite  iut-paris13
a2ensite  univ-paris13
```

Orange Restricted

Pour la désactivation des sites :

```
a2dissite  iut-paris13
a2dissite  univ-paris13
```

4.4. Adaptation du fichier /etc/hosts

Pour la suite, il faut ajouter la résolution DNS statique au niveau des fichiers hosts des différentes machines clientes :

```
192.168.1.100 www.iut-paris13.fr
192.168.1.100 www.univ-paris13.fr
```

4.5. Relance serveur et tests

Relancer votre serveur Apache2 pour la prise en compte de la nouvelle configuration :

```
> /etc/init.d/apache2 restart
ou
> service apache2 restart
```

Ouvrez votre navigateur et testez en tapant dans la barre d'adresse:
http://www.iut-paris13.fr
http://www.univ-paris13.fr
Vous devriez voir apparaître les pages d'accueil des deux sites que vous avez crée précédemment.

5. Sécurisation des accès – Fichier .htaccess

5.1. Sécurisation par authentification

a. Création des différents utilisateurs

On va commencer par créer 3 utilisateurs (avec leurs mots de passe respectifs chiffrés) dans le fichier httpUtilisateurs. Donnez l'autorisation à 2 des utilisateurs sur le fichier .htaccess et aucune pour le 3ème utilisateur.
Pour cela, utilisez la commande htpasswd pour créer des utilisateurs et affecter un mot de passe chiffré pour chaque utilisateur.
Exemple :

```
[3 root@WebServer /etc/apache2]$ htpasswd -c httpUtilisateurs Dupont
New password:
Re-type new password:
Adding password for user Dupont
[3 root@WebServer /etc/apache2]$ htpasswd  httpUtilisateurs martin
New password:
Re-type new password:
Adding password for user martin
[0 root@WebServer /etc/apache2]$
```

b. Création du fichier .htaccess

Nous devons créer le fichier .htaccess que l'on placera dans le répertoire à protéger (/var/www/univ-paris13).

```
AuthType Basic
AuthName "L'authentification est nécéssaire pour l'accès. Merci de fournir le login + mdp"
AuthUserFile /etc/apache2/httpUtilisateurs
Require User Dupont Martin
```

Vous constatez que le fichier contenant les utilisateurs et leur mot de passe se situe dans /etc/apache2/httpUtilisateurs.
Seuls les utilisateurs Dupont et Martin auront accès au VirtualHost /var/www/univ-paris13

Orange Restricted

c. *configuration du virtualhost*

Modifier la configuration du virtualhost pour appliquer les directives contenues dans le fichier .htaccess. Pour cela ajouter : ***AllowOverride AuthConfig***

La configuration de votre virtualhost ressemblerait à ceci :

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        ServerName www.univ-paris13.fr
        ServerAlias univ-paris13
        DocumentRoot /var/www/univ-paris13
        DirectoryIndex  univ.php

       <Directory /var/www/univ-paris13>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride AuthConfig
                Order allow,deny
                allow from all
        </Directory>

        ErrorLog /var/log/apache2/error.log

        # Possible values include: debug, info, notice, warn, error, crit,
        # alert, emerg.
        LogLevel warn

        CustomLog /var/log/apache2/access.log combined
```

d. *Tests d'accès*

A partir des machines clients (m1 et m2) essayer d'accès à votre page web avec un utilisateur légitime (Dupont ou Martin) et un autre utilisateur non autorisé à accéder. Qu'observez-vous?

e. *Man in the middle*

Maintenant positionnez-vous au niveau de la machine Hacker et prenez des traces en observant les différents échanges entre les machines clientes et le serveur.

Quelle information le hacker pourrait intercepter? Peut-il intercepter des information secrete? Démontrer cela avec la trace prise.

Avec ce que le hacker a intercepté comme information, pourrait-il accéder à la page web protégée en passant pour un utilisateur légitime? Faite le test.

5.2. Htaccess – Restriction des adresses IP

Il y a deux méthodes pour procéder à une restriction d'accès IP :

- Refuser l'accès à certaines adresses IP
- Autoriser uniquement l'accès à certaines adresses IP

a. *Autoriser l'accès uniquement à m1 :*

Commençons tout d'abord par n'autoriser l'accès qu'à la machine m1 :

```
# ordre d'execution: refuser puis autoriser
Order Deny,Allow
# On n'autorise personne à accéder au site..
Deny from all
#...sauf l'addresse IP suivante
Allow from 192.168.1.1
```

Valider que la restriction est fonctionnelle et que l'accès est autoriser uniquement à partir de la machine m1 et que les 2 autres machines (m2 et hacker) ne peuvent pas y accéder.

b. *Interdire l'accès au hacker*

Maintenant que le hacker est bien identifié, on va autoriser l'accès à tout le monde sauf à l'adresse IP du hacker

Orange Restricted

```
# ordre d'execution: Autoriser puis refuser
Order Allow,Deny
#Autoriser tt le monde à accéder au site...
Allow from all
# sauf l'adresse IP
Deny from 192.168.1.10
```

Valider que la restriction est fonctionnelle et que le Hacker ne pourrait plus accéder à votre page web.
Valider que l'accès au serveur web a été refusé au Hacker dans les logs :

```
tail -t /val/log/apache2/error.log
```

Cette configuration est-elle suffisante pour la sécurité des échanges? Expliquez pourquoi?

6. SSL

L'objectif de cette partie et d'activer le support SSL. Pour accepter les requêtes SSL, Apache a besoin de deux fichiers : une clé pour le serveur et un certificat signé.
La signature de ce certificat est réalisée par un organisme de certification tiers (tel que Verisign ou Thawte). Cependant vous pouvez signez-vous même votre certificat, la seule différence sera un avertissement par le navigateur lors de l'accès à une ressource SSL de votre serveur.

6.1. *Activation du module SSL*

Pour que le protocole SSL puisse fonctionner avec le Serveur HTTP Apache2, il faut activer le module ssl:

```
sudo a2enmod ssl
```

Puis recharger la configuration d'Apache2 faites :

```
sudo service apache2 force-reload
```

6.2. Création du certificat aut-signé

On peut créer son certificat SSL auto signé avec openssl :

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:1024 -out
/etc/apache2/server.crt -keyout /etc/apache2/server.key
```

Explications :

- -x509 -nodes donne le type de certificat voulu
- -days 365 indique la durée de validité (en jours) de votre certificat
- -newkey rsa:1024 demande une clé RSA de 1024 bits - d'après la doc apache, il est déconseillé de créer une clé plus grosse pour des histoires de compatibilité
- -out /etc/apache2/server.crt est le chemin de votre certificat
- -keyout /etc/apache2/server.key est le chemin de la clé privée

Répondez alors aux questions posées et puis sécurisez l'accès au fichier de clés privés :

```
sudo chmod o-rw /etc/apache2/server.key
```

6.3. Ajout port d'écoute 443

Régler le serveur pour qu'il écoute (aussi) sur le port 443
Par défaut, Apache2 est configuré pour écouter sur le port 80. Vous pouvez vérifier les ports actifs sur votre machine à l'aide de la commande :

```
netstat -nlt
```

Orange Restricted

Or le protocole SSL a besoin d'émettre sur un port spécifique pour pouvoir fonctionner, celui qui est adopté en général est le port 443. Pour cela ajoutez la directive suivante dans le fichier /etc/apache2/ports.conf (sauf si celle-ci est déjà présente) :

```
Listen 443
```

Redémarrez Apache et vérifiez si le serveur écoute bien sur le port 443.

6.4. Création du fichier de configuration

Créez un nouveau virtualhost (iut-paris par exemple):

```
<VirtualHost *:443>
  ServerName www.iut-paris.fr
  DocumentRoot /var/www/iut-paris
        SSLEngine on
        SSLCertificateFile /etc/apache2/server.crt
        SSLCertificateKeyFile /etc/apache2/server.key
</VirtualHost>
```

N'oubliez pas de créer le répertoire de la page web et d'ajouter une page web (html ou php)
Ensuite, on active la configuration

```
a2ensite iut-paris
```

On relance apache

```
service apache2 restart
```

6.5. Tests et navigation https

D'abord commencer pour faire un test local sur le serveur web :

```
openssl s_client -connect localhost:443
```

Faite un test à partir des 2 machines clientes (m1 et m2) et valider que vous utilisez bien du SSL pour votre navigation web.

6.6. Ecoute réseau

Refaite la même chose que précédemment en se positionnant au niveau de la machine du hacker et essayer d'écouter le trafic entre la machine client et serveur web.
Qu'observez-vous? Détaillez le changement par rapport à l'écoute précédente?

6.7. Redirection http vers https

Maintenant que votre site web supporte https, vous avez la possibilité de faire la redirection du trafic http vers https. Pour faire changer la configuration de votre virtualhost en conséquence :

```
Redirection http vers https
<VirtualHost *:80>
    ServerName www.iut-paris.fr/
        Redirect / https://www.iut-paris.fr/
 </VirtualHost>
```

Validez que la redirection https est fonctionnelle.

Merci

Orange Restricted