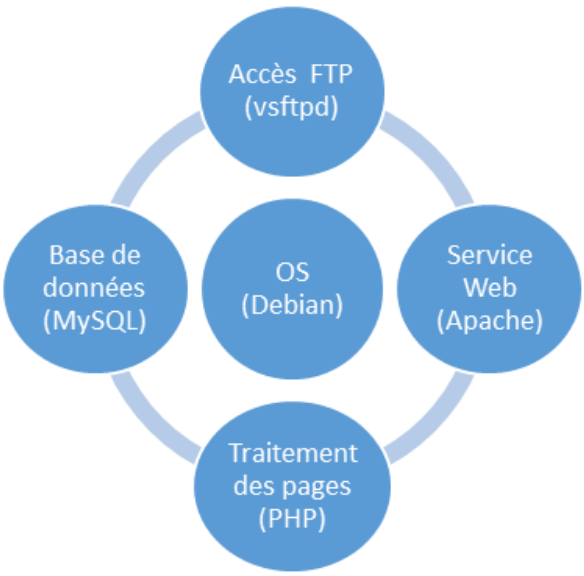
TP2 - Déploiement d'un serveur web Apache dans une machine virtuelle

L'objectif de ce TP est d'utiliser une machine virtuelle pour y installer un serveur web. Après une installation basique d'un serveur web Apache, avec une base de données et le support du langage de scripts PHP, nous verrons quelques éléments de configuration, ainsi que quelques problèmes de sécurité (sans les corriger). Lors de ce TP, vous écrirez directement les fichiers PHP dans le serveur, ce qui n'est pas la pratique habituelle. Un accès FTP est typiquement utilisé pour déployer une application web sur un serveur de production, que ce soit une vraie machine, une machine virtuelle ou encore chez un hébergeur. Une alternative est d'utiliser un Serveur Git à l'image de ce qui est vu dans le TP2 de services réseaux. Seules les grandes lignes des commandes seront décrites, pour avoir la syntaxe complète d'une commande on vous invite à utiliser le manuel : `man [commande]` *([ ] indique que c'est optionnel), exemple :* `man ls`*.*

IMPORTANT : changer le mot de passe de root ! !

1 Architecture

Comme indiqué ci-dessus, nous allons installer une base de données, le langage de scripts PHP et un service web permettant de traiter les requêtes HTTP qui seront réceptionnées. La seule chose que nous n'installerons pas et qui est souvent utilisé dans le cadre d'un serveur web mutualisé, comme le montre la figure ci-dessous, est un accès ftp.



Dans la suite nous allons donc installer les différents services, faire un peu de configuration et créer ce qu'il faut pour héberger deux sites web accessibles via des noms symboliques, à savoir `www.site1.fr` et `www.site2.fr`.

2 Installation

ATTENTION 1 : en dehors du navigateur qui devra être lancé (et installé si besoin) dans la machine hôte (la vraie machine). Les manipulations seront, sauf précision contraire, à faire dans la machine virtuelle avec le compte *root*.

ATTENTION 2 : lors de ce TP on utilise peut être une machine virtuelle avec un système en mode graphique avec un bureau, mais ce n'est pas ce qui est fait habituellement. Une machine faisant exclusivement office de serveur est généralement installée en mode texte et administrée à distance via SSH.

Les manipulations à effectuer sont :

1. quelles sont les adresses IP attribuées à la machine virtuelle ?
2. mettre à jour la liste des paquets disponibles, puis si des paquets peuvent être mis à jour lancer cette dernière

   ```
   apt update
   apt upgrade
   apt autoremove
   ```

3. on installe le service web, à savoir `apache`, ainsi que `php`

   ```
   apt update
   apt install apache2 php8.4
   ```

4. on installe un service de base de données, plus les fonctions qui permettront (si besoin) à `php` de communiquer avec celle(s)-ci. Dans l'immédiat on ne va pas configurer de base de données.

   ```
   apt install mariadb-server php8.4-mysql
   ```

5. installer le paquet `phpmyadmin` ;
6. avec le navigateur sur la vraie machine se connecter sur la machine virtuelle et vérifier que vous obtenez bien l'affichage de la page par défaut du serveur Apache, idem avec phpMyAdmin.

3 Configuration

3.1 Création des répertoires pour les fichiers des deux sites

Lors de son installation, la page par défaut qu'Apache affiche se trouve dans `/var/www/html` : c'est le fichier `index.html`. Pour ce premier TP sur l'installation d'un serveur web, nous n'allons pas faire nécessairement tout ce qu'il faut pour éviter tous les problèmes (notamment de sécurité). L'idée c'est déjà d'avoir quelque chose de fonctionnel et de découvrir quelques éléments de configuration.

Les manipulations à effectuer sont :

1. construction de l'arborescence

   ```
   mkdir -p -v /var/www/default
   mkdir -p -v /var/www/site1
   mkdir -p -v /var/www/site2
   ```

2. on créé un petit fichier dans `default`

```
cd /var/www/default
echo 'Un petit fichier' > my_file
```

3. créer dans le *home directory* de l'utilisateur *tpsae* sur la machine virtuelle le fichier `index.php` suivant :

```
<html>
<body>
<?php
  echo "Welcome on default page";
?>
</body>
</html>
```

3.2 Découverte de la "configuration" d'Apache

La documentation en ligne d'Apache est disponible via le lien

`https://httpd.apache.org/docs/2.4/`

Les manipulations à effectuer sont :

1. consulter le contenu du répertoire `/etc/apache2` ;
2. consulter plus en détail le fichier `apache2.conf` ;
3. afficher la liste des modules qui sont chargés via `apachectl -M` ;
4. l'affichage précédent peut éventuellement débuter avec un message d'erreur. Pour l'enlever, si besoin, ajouter à la fin du fichier `apache2.conf` la ligne suivante

   `ServerName localhost`

5. le fichier `/etc/apache2/sites-available/000-default.conf` contient la configuration du site par défaut. Modifier le fichier pour que `DocumentRoot` pointe sur `/var/www/default`. Ce fichier correspond à une configuration de type Serveur Virtuel par-Nom ou encore *VirtualHost*, on utilisera aussi ce type pour les deux sites ;
6. regarder le contenu de `/etc/apache2/sites-enabled`. Quel est le rôle de ce répertoire à votre avis ?
7. redémarrer le service HTTP, puis regarder son statut

   `systemctl restart apache2`

   ou

   `service apache2 restart`

   puis

   `systemctl status apache2`

   Connectez-vous à la machine virtuelle (ou recharger la page par défaut) avec le navigateur depuis la vraie machine. Que constatez-vous ? Est-ce une situation acceptable du point de vue de la sécurité ? Dans l'immédiat on ne va pas corriger cela ;
8. copier le fichier `index.php` dans le répertoire `/var/www/default` et recharger la page.

3.3 Configuration des sites www.site1.fr et www.site2.fr

Les manipulations à effectuer sont :

1. copier le fichier `index.php` dans chacun des deux répertoires (`site1` et `site2`) et le modifier pour que `default page` soit remplacé par `Site 1 de la machine de` $<$ *votre_nom* $>$ et `Site 2 de la machine de` $<$ *votre_nom* $>$ (vous remplacerez $<$ *votre_nom* $>$ avec le votre naturellement), respectivement ;
2. on passe ensuite à la création des deux sites au niveau d'Apache. Cela se fait sous la forme d'un fichier de configuration dans `/etc/apache2/sites-available`. Voici, ci-dessous, le fichier `site1.conf` pour le premier site (`www.site1.fr`). Vous vous baserez sur celui-ci pour définir la configuration du second site.

   ```
   <VirtualHost *:80>
     ServerAdmin webmaster@localhost
     ServerName www.site1.fr
     DocumentRoot /var/www/site1
     Alias /database "/usr/share/phpmyadmin/"
     ErrorLog ${APACHE_LOG_DIR}/error_site1.log
     CustomLog ${APACHE_LOG_DIR}/access_site1.log common
   </VirtualHost>
   ```

3. on active les deux sites via `a2ensite site1` et `a2ensite site2` (regarder ce qui apparaît dans le répertoire `/etc/apache2/sites-enabled`) ;
4. recharger Apache avec la commande `systemctl reload apache2`. Quelle est la différence avec `restart` ?
5. avec le navigateur sur la vraie machine se connecter sur `http://www.site1.fr`. Est-ce votre site ? Que faut-il mettre dans le fichier `/etc/hosts` de la machine hôte (la vraie) pour pouvoir se connecter ?
6. que constatez-vous après avoir fait la modification et une nouvelle tentative de connexion ? Quel peut être la source de ce problème ?
7. faire en sorte que vous puissiez également vous connecter sur le second site ;
8. désactiver le site par défaut (quelle est la commande inverse de `a2ensite` ?), sur quel site pointe l'adresse IP maintenant ?
9. est-il possible d'accéder à phpMyAdmin au niveau des deux sites ? Quelle façon d'y accéder est déclaré dans les fichiers `.conf` ?
10. PHP peut être utilisable sans restriction, ce qui peut poser de nombreux problèmes. À titre d'exemple, saisir le fichier `lire.php` ci-dessous dans le répertoire du site 1, puis le lire avec le navigateur

    ```
    <?php
      $list=file_get_contents('/etc/passwd');
      echo $list;
    ?>
    ```

    Comme vous devriez pouvoir le constater ou non, il se pourrait qu'il soit possible de lire un fichier du système qui est accessible par tout le monde sur le système (le fichier `passwd` dans cet exemple, mais ce n'est pas le seul...) ;

Enfin, comment pourrait-on utiliser la redirection de port vue dans le TP1 pour permettre à quelqu'un sur une autre machine d'accéder à vos sites sur votre machine virtuelle ?

4 Activation de TLS

Ici il s'agit de protéger les communications via l'utilisation d'un chiffrement fort en créant un certificat auto-signé pour chacun des sites. Naturellement, on redirigera les connexions qui arriveront sur le port `80`, non sécurisé, vers le port `443` qui le sera.

La documentation en ligne d'Apache est disponible via le lien

`https://httpd.apache.org/docs/2.4/ssl/`

Les manipulations à effectuer sont :

1. activation du module `ssl`
   (a) `a2enmod ssl`
   (b) on redémarre Apache
   ```
   systemctl restart apache2
   ```
2. génération d'un certificat auto-signé
   (a) on crée un répertoire pour y mettre sa clé privée et le certificat
   ```
   mkdir -p -v /etc/apache2/ssl/site1
   ```
   (b) on génère la clé privée et une requête de certificat pour le site `www.site1.fr`
   ```
   cd /etc/apache2/ssl/site1
   openssl genrsa -out site1.pem 3072
   openssl req -nodes -new -key site1.pem -out site1.csr
   ```
   concernant les informations demandées (`X` est toujours le numéro de votre machine) :
   — Pays → `FR` ;
   — Etat ou province → `FC` ;
   — Ville ou localité → `Belfort` ;
   — Organisation → `IUT-X` ;
   — Unité → `Info-X` ;
   — Common Name (FQDN) → `www.site1.fr` ;
   — Email → rien.
   Laisser les `'extra' attributes` vides. Ces informations pourraient être définies via un fichier de configuration.
   (c) finalement on génère un certificat auto-signé valide pour une année
   ```
   openssl x509 -req -days 365 -in site1.csr -signkey site1.pem -out site1.crt
   ```
   (d) refaire les étapes précédentes pour le second site.
3. on modifie la configuration des deux sites. Pour chacun on va ajouter une redirection de `http` vers `https`, soit du port `80` vers le port `443`. On ajoute également les premières directives pour activer TLS.

   Seules les indications pour `www.site1.fr` sont données, pour le second site il faut s'en inspirer.

   On rappelle que le fichier de configuration concernant le `virtualHost` pour `www.site1.fr` est le fichier `site1.conf` dans `/etc/apache2/sites-available`.

```
<VirtualHost *:80>
  ServerAdmin webmaster@localhost
  ServerName www.site1.fr
```

```
  Redirect permanent / https://www.site1.fr/
</VirtualHost>

<VirtualHost *:443>
  ServerAdmin webmaster@localhost
  ServerName www.site1.fr
  DocumentRoot /var/www/site1
  Alias /database "/usr/share/phpmyadmin/"
  ErrorLog ${APACHE_LOG_DIR}/error_site1.log
  CustomLog ${APACHE_LOG_DIR}/access_site1.log common

  SSLEngine on

  SSLCertificateFile /etc/apache2/ssl/site1/site1.crt
  SSLCertificateKeyFile /etc/apache2/ssl/site1/site1.pem

  <Directory /var/www/site1>
    Options -Indexes -FollowSymLinks +MultiViews
    AllowOverride none
    Require all granted
  </Directory>
</VirtualHost>
```

4. on affine la configuration du chiffrement en ajoutant des directives à la suite de la directive `SSLEngine on`
   (a) désactivation de tous les protocoles, puis activation de la version 1.3 de TLS
   ```
   SSLProtocol -All +TLSv1.3
   ```
   (b) sélection de méthodes de chiffrement fort
   ```
   SSLCipherSuite HIGH:!aNULL:!MD5:!ADH:!RC4:!DH
   ```
   (c) on force le navigateur à respecter l'ordre que l'on a précisé précédemment
   ```
   SSLHonorCipherOrder on
   ```
   (d) directives à utiliser en cas de certificat lié à une autorité de certification. Comme on en a pas, ne pas les mettre dans les fichiers de configuration. C'est uniquement un exemple.
   ```
   # Certificat racine si besoin
   SSLCACertificateFile /etc/apache2/ssl/site1/certif-racine.crt
   # Certificat intermediaire
   SSLCACertificateFile /etc/apache2/ssl/site1/certif-interm.crt
   ```

5. se connecter sur `www.site1.fr` et vérifier que la bascule d'HTTP vers HTTPS est fonctionnelle, avec utilisation du certificat créé précédemment (refaire le test avec l'autre site).

5 Problèmes de sécurité

Vous avez déjà vu à la fin de la section 3.3 qu'il était possible d'accéder à des informations du système via PHP. Nous allons revenir sur cet aspect et l'approfondir. Nous ne verrons pas systématiquement de solution pour résoudre chaque problème.

Les manipulations à effectuer sont :

1. éditer le fichier `shell.php` comme suit sur la machine hôte

   ```
   <?php
     $resultat=shell_exec('pwd');
     echo $resultat;
   ?>
   ```

2. l'uploader dans le site 1 , soit dans `/var/www/site1` ;
3. le lire dans le navigateur ;
4. modifier le script pour afficher le contenu du répertoire `/var/www/site2` ;
5. finalement afficher, toujours en modifiant le script, le contenu du fichier `index.php` dans le site 2 via `cat`. Vous afficherez le code source de la page via `View Page Source`.

Supposons maintenant que l'utilisateur ajoute un fichier dont il protège la lecture via un `.htaccess` dans le site 2. Pour cela il procède comme décrit ci-dessous.

Les manipulations à effectuer sont :

1. éditer le fichier `restreint.php` comme suit sur la machine hôte

   ```
   <?php
     echo "Je suis le fichier avec un acces restreint... normalement";
   ?>
   ```

2. l'uploader dans le site 2 ;
3. afin de créer le fichier contenant le mot de passe permettant de contrôler l'accès au fichier, nous aurons besoin de la commande `htpasswd` sur la machine hôte. Vérifier qu'elle est installée, sinon procéder comme suit :

   ```
   apt install apache2-utils
   ```

4. créer le fichier `.htpasswd` sur la machine hôte, celui-ci nous servira à contrôler l'accès au fichier `restreint.php`

   ```
   htpasswd -c .htpasswd tpsae
   ```

   il vous faudra donner un mot de passe. À noter que l'option `-c` doit être omise si on veut ajouter un autre utilisateur ;

5. éditer un fichier `.htaccess` sur la machine hôte avec le contenu suivant :

   ```
   AuthUserFile /var/www/site2/.htpasswd
   AuthType Basic
   AuthName "Protected file"
   <Files "restreint.php">
     Require valid-user
   </Files>
   ```

6. uploader les deux fichiers dans le répertoire où se trouve le fichier `restreint.php` ;

7. modifier le fichier de configuration `site2.conf` en ajoutant après `CustomLog` les lignes suivantes :

```
<Directory /var/www/site2>
    Options -Indexes -FollowSymLinks +MultiViews
    AllowOverride AuthConfig
    Require all granted
</Directory>
```

8. redémarrer le service web ;
9. puis essayer d'accéder au fichier `restreint.php` grâce au navigateur via `www.site2.fr` et constater qu'une fenêtre contrôlant l'accès s'ouvre ;
10. modifier le script `shell.php` pour qu'il affiche le contenu de `restreint.php` ;
11. que constatez-vous en lisant le fichier `restreint.php` via `www.site1.fr` avec ce script ?
    Vous afficherez le code source de la page via `View Page Source`.

Pour éviter les affichages produits par `shell.php`, nous allons jouer sur deux éléments : d'une part restreindre la portée des fichiers atteignables via PHP et d'autre part désactiver la fonction `shell_exec`.

Les manipulations à effectuer sont :

1. ajouter la directive suivante dans le fichier de configuration du site 2, juste avant `Options` dans le bloc `Directory`

   ```
   php_admin_value open_basedir "/var/www/site2/"
   ```

   puis redémarrer le service web ;
2. fermer et relancer le navigateur, puis relire le fichier et constater le changement. On voit donc que cette directive restreint la portée des fichiers atteignables via PHP. Le fait que le paramètre `php_admin_value` soit dans la configuration du `virtualHost` permet de régler finement les droits, contrairement à un réglage global dans un fichier de configuration de PHP ;
3. justement, pour désactiver la fonction `shell_exec`, c'est au niveau du fichier de configuration globale qu'il faut intervenir. Il faut ainsi ajouter la fonction dans la liste de celles qui sont désactivées (`disable_functions`) dans le fichier suivant :

   `/etc/php/8.4/apache2/php.ini`

6 *Web Application Firewall*

Nous allons maintenant installer `ModSecurity`, un module open source qui permet d'accroître fortement la sécurité d'un serveur web Apache. Il permet d'assurer un certain niveau de protection contre de nombreuses attaques exploitant des failles web : XSS, injection SQL, vol de session, etc. (consulter le site web `https://github.com/SpiderLabs/ModSecurity`. Après son installation, on fera quelques tests pour vérifier son bon fonctionnement.

6.1 Installation

Les manipulations à effectuer sont :

1. on installe le module

```
apt install libapache2-mod-security2
```

2. on vérifie que le module est bien installé et activé

   ```
   apachectl -M | grep security2
   ```

3. par défaut le module ne fonctionne pas. Il faut d'abord le configurer

   ```
   mv /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf
   ```

   puis on édite le fichier `.conf` et on met `SecRuleEngine On`, avant de redémarrer Apache avec `systemctl restart apache2` ;

4. le module a besoin de règles pour fonctionner. Des règles par défaut ont été mises en place, mais on va récupérer celles qui sont accessibles via `GitHub`

   (a) on supprime les règles qui ont été installées

   ```
   rm -rf /usr/share/modsecurity-crs
   ```

   (b) on télécharge les dernières règles

   ```
   apt install git
   git clone https://github.com/SpiderLabs/owasp-modsecurity-crs.git /usr/share/modsecurity-crs
   ```

   (c) on renomme le fichier

   ```
   cd /usr/share/modsecurity-crs
   mv crs-setup.conf.example crs-setup.conf
   ```

   (d) on met à jour la configuration du module dans Apache en modifiant le fichier `/etc/apache2/mods-enabled/security2.conf` comme suit

   ```
   IncludeOptional /etc/modsecurity/*.conf
   IncludeOptional /usr/share/modsecurity-crs/*.conf
   IncludeOptional /usr/share/modsecurity-crs/rules/*.conf
   ```

   et on met en commentaire la ligne `IncludeOptional` concernant OWASP ;

   (e) on redémarre Apache.

6.2 Tests

Pour illustrer le bon fonctionnement du module et son intérêt, nous allons envoyer des requêtes représentatives de deux types d'attaques, à savoir l'injection d'un script (attaque de type XSS) et une injection SQL.

Les manipulations à effectuer sont :

1. saisir au niveau du navigateur

   ```
   https://www.site1.fr/?q="><script>alert(1)</script>
   ```

2. quel est le message affiché au niveau des logs (fichier `modsec_audit.log`) ?
3. faire la même manipulation avec

   ```
   https://www.site1.fr/?q=1' OR 1=1
   ```

4. désactiver le module avec `a2dismod security2`, redémarrer Apache, puis refaire les manipulations précédentes et constater la différence.

7 Redirection de port pour accéder au sites de la VM

Lors du TP1, vous avez vu comment rediriger le port 5900 du service VNC qui permettait de prendre le contrôle de la session graphique courante (`x11vnc`) depuis une autre machine de la salle de TP. En effet, le port 5900 avait été redirigé vers le port 2026 de la machine hôte.

L'idée est de faire de même en redirigeant le port 80 de la VM sur le port 8080 de la machine hôte. Vous regarderez en particulier quel est le site qui est redirigé. C'est-à-dire quel site s'affiche en entrant par exemple l'adresse `http://localhost:8080` dans un navigateur de la machine hôte. Vous essaierez ensuite de vous connecter sur le site web d'un(e) voisin(e) en utilisant l'adresse IP de sa machine à la place de `localhost`.