TP1 : Installation et administration d'un serveur web

Préambule : *Administration d'un serveur web sécurisé, avec Apache.*

Consignes pour le TP :

- **Rédiger seul·e ou en binôme un rapport tout au long de la séance**, qui contient les configurations et commandes utilisées, vos explications et vos réponses aux éventuelles questions. Ce rapport au format texte brut ou PDF, devra être envoyé par mail à l'intervenant·e, avec le sujet suivant : « *[SSSR][TP1] Nom1 Nom2* ».

- **Remettre la machine en état avant de partir**, en rétablissant la configuration par défaut du réseau, en désinstallant les services inutiles, et en la redémarrant. **Penser à copier auparavant les fichiers** qui ont été modifiés/générés au fil des questions.

- Hormis pour utiliser Firefox et Wireshark, tout le TP doit exclusivement se faire **en utilisant des commandes non-graphiques**, dans un ou plusieurs terminaux (permettant ainsi de se rapprocher des conditions réelles de l'administration de serveurs). Utiliser *vim* ou *emacs* pour éditer les fichiers de configuration est fortement conseillé (en prenant un peu de temps pour suivre un tutoriel, à la fois pour comprendre comment les utiliser, et saisir pourquoi tant de professionnel·le·s ne jurent que par leur utilisation).

Partie I : Configuration d'un serveur web et ses vhosts

1. Apache est le serveur web (i.e. un serveur interrogeable en HTTP / HTTPS) le plus populaire dans le monde, bien qu'il soit de plus en plus concurrencé par de nouveaux serveurs plus performants (e.g. Nginx). Installer Apache en installant le paquet *apache2*. Vérifier sa bonne installation en consultant votre premier site web : http://localhost/.

2. Les fichiers de configuration d'Apache sont stockés dans */etc/apache2/*. La documentation officielle est disponible à l'adresse suivante : http://httpd.apache.org/docs/current/.

3. Pour la suite de ce TP, il faut avoir à sa disposition un nom de domaine, afin de réaliser les tests. Demander à votre binôme de compléter le fichier */etc/hosts* de sa propre machine avec votre adresse IPv6[1], ainsi que le nom de domaine de votre choix (e.g. *mylittle.pony*, qui sera pris **en exemple** dans la suite du sujet).

4. Le répertoire *sites-available/* contient les sites virtuels (les vhosts) hébergés. Sur votre machine, copier le fichier de configuration du site par

1 Si la machine est dans un réseau sans IPv6, tout le monde ajoute une adresse générée par http://unique-local-ipv6.com à son interface *eth0* (nous ne pourrons malheureusement pas utiliser les adresses *fe80::/8*, à cause de Firefox).

défaut (*000-default.conf*), en nommant la copie *mylittlepony.conf.* Dans ce fichier, modifier la variable « *ServerName* » avec *mylittle.pony* et la variable « *DocumentRoot* » avec */var/www/mylittlepony/*[2] (décommenter les variables, si nécessaire). Enfin, activer le site avec la commande : *a2ensite mylittlepony* et relancer (le service de) Apache[3].

5. Créer un fichier *index.html* dans le dossier */var/www/mylittlepony/*, avec le contenu de votre choix. Vérifier que votre binôme est capable d'accéder à votre superbe site web, en saisissant l'adresse *mylittle.pony* dans son propre navigateur.

6. Apache a une architecture modulaire qui permet de lui ajouter des fonctionnalités. Installer le paquet *libapache2-mod-php5*. Créer le fichier PHP suivant dans */var/www/mylittlepony/test.php* :

```
My IP address is: <?= $_SERVER['REMOTE_ADDR'] ?>
```

7. Vérifier, grâce à votre binôme, si votre page PHP affiche bien son adresse IPv6. Lui demander maintenant de modifier son fichier */etc/hosts* pour remplacer votre adresse IPv6 par votre adresse IPv4, et de recharger la page PHP. Expliquer quelle est la logique qui a conduit son ordinateur à utiliser une adresse IP différente pour afficher votre site.

8. Créer un second site, avec un nom de domaine différent, et vérifier que les deux sont bien accessibles en même temps. Demander ensuite à votre binôme de faire pointer un troisième nom de domaine, de son choix, vers votre ordinateur et de tenter d'y accéder avec son navigateur.

   a) Que se passe-t-il ?
   b) Désactiver le site par défaut avec la commande *a2dissite 000-default* et relancer Apache. En retentant un accès avec le troisième nom de domaine, que se passe-t-il désormais ?
   c) Que peut-on en déduire, en termes de sécurité ?

Partie II : HTTPS et gestion des certificats

1. Modifier la configuration du vhost de *mylittle.pony*, pour ajouter une authentification par utilisateur + mot de passe sur votre site, en ajoutant à la fin du fichier (mais **toujours** dans l'arborescence de <*VirtualHost*>) :

```
<Location ~>
  AuthType Basic
  AuthName "Authentication Required"
  AuthUserFile "/etc/apache2/htpasswd/mylittlepony"
  Require valid-user

  Order allow,deny
  Allow from all
</Location>
```

2 Ici et pour le reste de ce TP, il faudra toujours veiller à créer les dossiers qui n'existeraient pas déjà.
3 Attention, il faut toujours penser à utiliser *sudo* pour relancer Apache.

2. Utiliser la commande *htpasswd -c /etc/apache2/htpasswd/mylittlepony toto* pour créer un utilisateur *toto*, en choisissant un mot de passe bidon (installer, si nécessaire, le paquet *apache2-utils*, pour pouvoir utiliser cette commande). Redémarrer Apache, et demander à votre binôme de tester l'authentification.

3. Utiliser Wireshark pendant que votre binôme recharge la page, pour prouver qu'il suffit de sniffer le réseau pour connaître le couple utilisateur + mot de passe utilisé (*astuce n°1 :* dans Wireshark, cliquer à droite sur un paquet à destination de HTTP, puis choisir « *Follow TCP Stream* » pour visualiser la requête HTTP complète – *astuce n°2 :* utiliser la commande *echo xxx | base64 -d* pour convertir un contenu *xxx* encodé en Base64 en un format lisible).

4. Dès lors qu'un site est consulté en HTTP, n'importe quel administrateur ou administratrice de n'importe lequel des ordinateurs intermédiaires utilisés pour atteindre la destination, est capable de savoir exactement ce que vous faites, en inspectant ce qu'on lui demande de faire transiter. Utiliser la commande *traceroute* pour avoir un ordre d'idée du nombre d'intermédiaires utilisés pour atteindre un site de votre choix (e.g. *traceroute lesjoiesdusysadmin.fr*).

5. Le protocole HTTPS fonctionne grâce à l'utilisation d'une paire de clés (certificats), l'une étant publique et l'autre privée. En accédant à un site en HTTPS, votre navigateur va utiliser la clé publique du site pour chiffrer un mot de passe secret, que seule la clé privée de votre site pourra déchiffrer. Il va ensuite envoyer ce message chiffré à votre serveur web. Votre serveur Apache doit être capable de déchiffrer le message, pour prendre connaissance de ce mot de passe secret, et l'utiliser pour chiffrer et déchiffrer le reste des communications avec votre navigateur. Créer une paire de clés pour votre site web, grâce à la commande *openssl*, en indiquant le nom de domaine utilisé pour accéder à votre site, dans le champ « *Common Name* » (appuyer simplement sur Entrée, pour les autres questions) :

```
openssl req -x509 -nodes -newkey rsa:2048 -keyout
/etc/apache2/ssl/mylittlepony.key -out
/etc/apache2/ssl/mylittlepony.crt
```

6. Activer le module SSL grâce à la commande *a2enmod ssl*, puis modifier le vhost de *mylittle.pony*, pour qu'il écoute sur le port 443 (HTTPS) plutôt que 80 (HTTP). Enfin, ajouter les lignes suivantes à la fin du fichier de configuration, pour que le vhost soit capable de répondre en HTTPS grâce aux certificats créés (celui finissant par *.key* correspondant à la clé secrète) :

```
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/mylittlepony.crt
SSLCertificateKeyFile /etc/apache2/ssl/mylittlepony.key
```

7. Afin d'être à jour des bonnes pratiques et d'avoir un site web réellement sécurisé en fonction des derniers exploits connus, ajouter les lignes de configuration telles que conseillées sur le site https://cipherli.st. Redémarrer Apache. Si le redémarrage échoue, ouvrir un autre terminal et lancer la commande « *tailf /var/log/apache2/error.log* » pour voir les erreurs en temps réel. Retenter de redémarrer Apache depuis le terminal

précédent, et prendre connaissance des nouvelles erreurs qui s'affichent, pour trouver un moyen de les corriger (*indice :* il y a probablement un module de plus à activer, et des commandes non-supportées par votre version de Apache, à supprimer).

8. Une fois que le service de Apache a correctement redémarré (le meilleur moyen de s'en assurer étant de saisir la commande *echo $?* juste après avoir faire le redémarrage du service, et de vérifier qu'on obtient bien *0*), accéder à *mylittle.pony* depuis l'ordinateur de votre binôme, en HTTPS (il faudra lever l'exception de sécurité du navigateur, qui n'apprécie pas de voir un certificat non-certifié). Démontrer grâce à Wireshark qu'il est désormais impossible pour un intermédiaire de voir l'utilisateur et le mot de passe utilisés (ni même le contenu de la page, qui ne le regarde pas).

9. En se documentant, expliquer simplement pour quelle raison l'intermédiaire n'est pas en mesure de déchiffrer les données, même s'il a intercepté tous les échanges entre le navigateur et le serveur web, dès le début des transactions.

Partie III : Quelques exercices d'administration

1. Votre site web accessible en HTTPS renvoit désormais la page par défaut de votre serveur, dès lors qu'on y accède en HTTP. Trouver une solution pour qu'il redirige automatiquement vers la version HTTPS (il existe au moins 3 solutions, en créant un second vhost, et en configurant la redirection HTTP grâce à HTML, PHP ou Apache lui-même).

2. Trouver un moyen de personnaliser la page qui indique que l'utilisateur et/ou le mot de passe saisis sont faux, lorsque l'utilisateur se trompe au moment de l'authentification.

3. **BONUS EXPERT·E·S :** Utiliser la commande *telnet* pour interroger votre serveur web (sans HTTPS) manuellement et en mode texte, sans l'aide d'un navigateur. Si vous êtes un·e champion·ne et que vous êtes arrivé·e jusqu'ici avant la fin du TP, se renseigner sur les nouveautés apportées par HTTP/2, qui devrait petit-à-petit remplacer HTTP 1.1.