Installation et sécurisation d'un serveur Apache

Notions Systèmes Réseaux

TP Chapitre 4.5

Sécuriser la configuration d’un serveur web

1. Configuration globale d'Apache2

1.1. Configuration d’Apache

Tous les fichiers de configuration d’apache sont regroupés dans le dossier /etc/apache2. Nous allons les détailler afin de mieux s’y retrouver.

- apache2.conf : Fichier principal de configuration
- conf-available : Contient la liste des configurations
- conf-enabled : Contient la liste des configurations activées
- envvars : Contient les variables d’environnement d’Apache
- magic : sert au module mime_magic (bibliothèque pour détecter le type mime d'un fichier par extension ou par son contenu)
- apache2.conf : Fichier de configuration principal appelant les autres
- mods-available : Contient la liste des modules installés
- mods-enabled : Contient la liste des modules utilisés
- ports.conf : Contient la liste des ports sur lesquels apache écoute
- sites-available : Contient le fichier de configuration des différents sites installés
- sites-enabled : Contient le fichier de configuration des différents sites utilisés

```
root@Predator:~# cd /etc/apache2/
root@Predator:/etc/apache2# ls -la
total 96
drwxr-xr-x   8 root root  4096 janv. 12 13:10 .
drwxr-xr-x 142 root root 12288 janv. 26 10:16 ..
-rw-r--r--   1 root root  7224 août  14 16:36 apache2.conf
drwxr-xr-x   2 root root  4096 janv. 12 13:10 conf-available
drwxr-xr-x   2 root root  4096 janv. 12 13:10 conf-enabled
-rw-r--r--   1 root root  1782 août  14 16:36 envvars
-rw-r--r--   1 root root 31063 août  14 16:36 magic
drwxr-xr-x   2 root root 12288 janv. 17 17:54 mods-available
drwxr-xr-x   2 root root  4096 janv. 12 13:14 mods-enabled
-rw-r--r--   1 root root   320 août  14 16:36 ports.conf
drwxr-xr-x   2 root root  4096 janv. 12 13:10 sites-available
drwxr-xr-x   2 root root  4096 janv. 12 13:10 sites-enabled
```

1.1.1. Parcourons-les !

Tout d'abord, il faut savoir qu'avant Apache2 il y avait Apache et qu'à peu près toutes les informations de configuration étaient dans un seul et unique fichier, **httpd.conf.**

Cela posait quelques problèmes, car ce fichier devenait un peu un fourre-tout dans lequel il était difficile de savoir où certaines informations étaient situées et s'il n'y avait pas des informations redondantes.

Pour Apache2, ce fichier de configuration a été séparé en plusieurs parties. Le point de départ est le fichier apache2.conf.

Ce fichier contient un certain nombre de directives importantes, ainsi que les inclusions (Includes) des autres fichiers de configuration.

Cependant, il n'y a pas de directives qui nous intéressent dans ce fichier pour l'instant. Par contre, les includes en fin de fichier nous donnent une idée des parties de la configuration qui sont en dehors de apache2.conf, et qui vont pouvoir nous intéresser :

```
# Include module configuration:
Include mods-enabled/*.load
Include mods-enabled/*.conf

# Include list of ports to listen on
Include ports.conf

#
# The following directives define some format nicknames for use with
# a CustomLog directive.
#
# These deviate from the Common Log Format definitions in that they use %O
# (the actual bytes sent including headers) instead of %b (the size of the
# requested file), because the latter makes it impossible to detect partial
# requests.
#
# Note that the use of %{X-Forwarded-For}i instead of %h is not recommended.
# Use mod_remoteip instead.
#
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

# Include of directories ignores editors' and dpkg's backup files,
# see README.Debian for details.

# Include generic snippets of statements
IncludeOptional conf-enabled/*.conf

# Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf
```

Nous voyons d'abord l'inclusion de modules.

En effet, Apache est un service modulaire. Cela veut dire que l'on peut lui ajouter des modules qui viennent lui ajouter des fonctionnalités particulières.

Par exemple, Apache peut jouer le rôle de proxy. Pour cela, il faut lui ajouter le module mod_proxy.

On peut ainsi ajouter toutes sortes de fonctionnalités à Apache.

Sans le savoir, nous avons déjà ajouté un module à Apache en installant libapache2-mod-php qui est le module de prise en compte du langage PHP par notre serveur Apache2.

Nous avons deux lignes includes pour les modules :

```
# Include module configuration:
Include mods-enabled/*.load
Include mods-enabled/*.conf
```

Ceci nous indique qu'Apache va prendre en compte les fichiers .load et .conf situés dans le répertoire mods-enabled (qui est lui-même dans /etc/apache2).

Allons donc voir ce qui est contenu dans le répertoire mods-enabled :

```
ls -la mods-enabled/
total 8
drwxr-xr-x 2 root root 4096 janv. 12 13:14 .
drwxr-xr-x 8 root root 4096 janv. 12 13:10 ..
lrwxrwxrwx 1 root root   36 janv. 12 13:10 access_compat.load ->
../mods-available/access_compat.load
lrwxrwxrwx 1 root root   28 janv. 12 13:10 alias.conf -> ../mods-available/alias.conf
lrwxrwxrwx 1 root root   28 janv. 12 13:10 alias.load -> ../mods-available/alias.load
lrwxrwxrwx 1 root root   33 janv. 12 13:10 auth_basic.load -> ../mods-available/auth_basic.load
lrwxrwxrwx 1 root root   33 janv. 12 13:10 authn_core.load -> ../mods-available/authn_core.load
lrwxrwxrwx 1 root root   33 janv. 12 13:10 authn_file.load -> ../mods-available/authn_file.load
lrwxrwxrwx 1 root root   33 janv. 12 13:10 authz_core.load -> ../mods-available/authz_core.load
lrwxrwxrwx 1 root root   33 janv. 12 13:10 authz_host.load -> ../mods-available/authz_host.load
lrwxrwxrwx 1 root root   33 janv. 12 13:10 authz_user.load -> ../mods-available/authz_user.load
lrwxrwxrwx 1 root root   32 janv. 12 13:10 autoindex.conf -> ../mods-available/autoindex.conf
lrwxrwxrwx 1 root root   32 janv. 12 13:10 autoindex.load -> ../mods-available/autoindex.load
lrwxrwxrwx 1 root root   30 janv. 12 13:10 deflate.conf -> ../mods-available/deflate.conf
lrwxrwxrwx 1 root root   30 janv. 12 13:10 deflate.load -> ../mods-available/deflate.load
lrwxrwxrwx 1 root root   26 janv. 12 13:10 dir.conf -> ../mods-available/dir.conf
lrwxrwxrwx 1 root root   26 janv. 12 13:10 dir.load -> ../mods-available/dir.load
lrwxrwxrwx 1 root root   26 janv. 12 13:10 env.load -> ../mods-available/env.load
lrwxrwxrwx 1 root root   29 janv. 12 13:10 filter.load -> ../mods-available/filter.load
lrwxrwxrwx 1 root root   27 janv. 12 13:10 mime.conf -> ../mods-available/mime.conf
lrwxrwxrwx 1 root root   27 janv. 12 13:10 mime.load -> ../mods-available/mime.load
lrwxrwxrwx 1 root root   34 janv. 12 13:14 mpm_prefork.conf ->
../mods-available/mpm_prefork.conf
lrwxrwxrwx 1 root root   34 janv. 12 13:14 mpm_prefork.load ->
../mods-available/mpm_prefork.load
lrwxrwxrwx 1 root root   34 janv. 12 13:10 negotiation.conf ->
../mods-available/negotiation.conf
lrwxrwxrwx 1 root root   34 janv. 12 13:10 negotiation.load ->
../mods-available/negotiation.load
lrwxrwxrwx 1 root root   29 janv. 12 13:14 php7.3.conf -> ../mods-available/php7.3.conf
lrwxrwxrwx 1 root root   29 janv. 12 13:14 php7.3.load -> ../mods-available/php7.3.load
lrwxrwxrwx 1 root root   33 janv. 12 13:10 reqtimeout.conf -> ../mods-available/reqtimeout.conf
lrwxrwxrwx 1 root root   33 janv. 12 13:10 reqtimeout.load -> ../mods-available/reqtimeout.load
lrwxrwxrwx 1 root root   31 janv. 12 13:10 setenvif.conf -> ../mods-available/setenvif.conf
lrwxrwxrwx 1 root root   31 janv. 12 13:10 setenvif.load -> ../mods-available/setenvif.load
lrwxrwxrwx 1 root root   29 janv. 12 13:10 status.conf -> ../mods-available/status.conf
lrwxrwxrwx 1 root root   29 janv. 12 13:10 status.load -> ../mods-available/status.load
```

Les plus avertis d'entre vous auront remarqué que tous ces fichiers ne sont que des liens vers d'autres fichiers qui sont situés dans ../mods-available/.

En fait, le répertoire mods-enabled ne contient que des liens. Ces liens pointent vers les vrais fichiers qui sont contenus dans mods-available.

Nous venons de découvrir le fonctionnement modulaire d'Apache2.

- Tous les modules sont installés dans mods-available.
- On crée des liens vers ces fichiers dans mods-enabled.
- Seul le répertoire mods-enabled est lu par la configuration d'Apache.

Ce mode de fonctionnement nous permettra très facilement de désactiver ou d'activer un module pour Apache2. Il suffira de créer un lien vers le module ou au contraire de l'effacer.

Vous pouvez faire cela à la main, ou utiliser la commande **a2enmod**.

Donc, pour installer un module, vous avez deux choix :

- l'installer avec la commande apt-get install nom_module ;
- récupérer les fichiers .conf et .load et les mettre dans le répertoire mods-available, puis créer des liens vers ces fichiers dans mods-enabled.

Vous savez donc maintenant installer, désactiver ou activer un module.

Nous allons passer à la suite des includes de notre fichier apache2.conf.

```
Include ports.conf
```

Ce fichier va indiquer sur quel port notre serveur doit écouter. Par défaut, il s'agit du port 80, mais cela peut être modifié si vous le souhaitez (mais vous ne voudriez pas outrepasser une norme, n'est-ce pas ?)

Il y a ensuite quelques informations sur le format des logs du service et le répertoire conf.d qui peut comprendre des attributs de configuration particuliers, mais cela ne nous intéresse pas ici.

Les logs sont des fichiers qui contiennent des informations écrites sur tout ce qui se passe au niveau de la machine. Les logs d'Apache indiqueront tout ce qui se passe au niveau du service, chaque connexion, chaque erreur, etc. C'est très utile pour trouver la source d'un problème.

Et enfin une dernière partie très importante qui concerne les virtualhosts :

```
# Include the virtual host configurations:
Include sites-enabled/
```

Le virtualhost est une notion très importante dans Apache2. C'est ce qui nous permet de faire tourner plusieurs sites sur le même serveur Apache2.

Vous pouvez avoir un serveur web qui héberge plusieurs sites web.

Nous allons donc regarder ce qui est contenu dans le répertoire sites-enabled.

```
root@Predator:/etc/apache2# ls -la sites-available/
total 20
drwxr-xr-x 2 root root 4096 janv. 12 13:10 .
```

```
drwxr-xr-x 8 root root 4096 janv. 12 13:10 ..
-rw-r--r-- 1 root root 1332 août  14 16:36 000-default.conf
-rw-r--r-- 1 root root 6338 août  14 16:36 default-ssl.conf
```

Un seul fichier ici ! Enfin, un lien plus exactement, mais nous connaissons le principe maintenant, qui est le même que pour les modules avec un répertoire sites-available qui contient les fichiers et le répertoire sites-enabled qui contient les liens vers les fichiers dans sites-available.

Ce fichier est le fichier de configuration par défaut de nos virtualhosts. C'est lui qui sera utilisé par défaut pour toute requête web arrivant à notre serveur.

Regardons son contenu :

```
root@Predator:/etc/apache2# cat sites-available/000-default.conf

<VirtualHost *:80>
	# The ServerName directive sets the request scheme, hostname and port that
	# the server uses to identify itself. This is used when creating
	# redirection URLs. In the context of virtual hosts, the ServerName
	# specifies what hostname must appear in the request's Host: header to
	# match this virtual host. For the default virtual host (this file) this
	# value is not decisive as it is used as a last resort host regardless.
	# However, you must set it for any further virtual host explicitly.
	#ServerName www.example.com
	ServerAdmin webmaster@localhost
	DocumentRoot /var/www/html

	# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
	# error, crit, alert, emerg.
	# It is also possible to configure the loglevel for particular
	# modules, e.g.
	#LogLevel info ssl:warn

	ErrorLog ${APACHE_LOG_DIR}/error.log
	CustomLog ${APACHE_LOG_DIR}/access.log combined

	# For most configuration files from conf-available/, which are
	# enabled or disabled at a global level, it is possible to
	# include a line for only one particular virtual host. For example the
	# following line enables the CGI configuration for this host only
	# after it has been globally disabled with "a2disconf".
	#Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

Nous n'avons besoin de comprendre que quelques éléments pour l'instant.

<VirtualHost *:80>

Ceci indique le début d'une directive de configuration Apache et notamment le début d'un virtualhost qui sera en écoute sur le port 80.

DocumentRoot /var/www/html

Le DocumentRoot est très important. C'est ce qui va indiquer au serveur Apache2 où vont se situer les pages de notre site web. Dans notre cas, comme dans la majorité des cas, la racine de notre site web se situera dans /var/www/html.

D'ailleurs, nous pouvons aller voir le contenu de ce répertoire :

```
root@Predator:/etc/apache2# ls -la /var/www
total 12
drwxr-xr-x  3 root  root 4096 janv. 12 13:10 .
drwxr-xr-x 15 root  root 4096 janv. 12 13:10 ..
drwxr-xr-x  2 roger root 4096 janv. 12 14:38 html
```

Il contient le répertoire html. Allons voir ce qu'il contient :

```
root@Predator:/etc/apache2# ls -la /var/www/html/
total 12
drwxr-xr-x 2 roger root  4096 janv. 27 21:28 .
drwxr-xr-x 3 root  root  4096 janv. 12 13:10 ..
-rw-r--r-- 1 roger roger  358 janv. 12 15:22 index.html
```

Il ne contient qu'un fichier index.html.

Nous pouvons essayer de faire une modification et de voir le résultat.

Éditez via VI ou VIM le fichier index.html et mettez-y... ce que vous voulez !

```
<html>
<body>
<h1>Mon nouveau titre pour un serveur Apache tout neuf !</h1>
<p>Bienvenue sur mon serveur Apache 2 !</p>
<p>Lorem Ipsum...</p>
</body>
</html>
```

Testez ensuite votre page dans le navigateur.

1.2. Exercice autour d'Apache

Nous allons voir ici quelques éléments de configuration du serveur Apache2 qui pourraient vous intéresser. Dans un premier temps, nous allons modifier la configuration et voir les conséquences de celles-ci.

1.2.1. Bidouillons gaiement !

Comme premier test, nous avons modifié le contenu de notre fichier index.html. Nous allons maintenant essayer de changer son nom pour voir les conséquences.

Nous allons l'appeler index.txt :

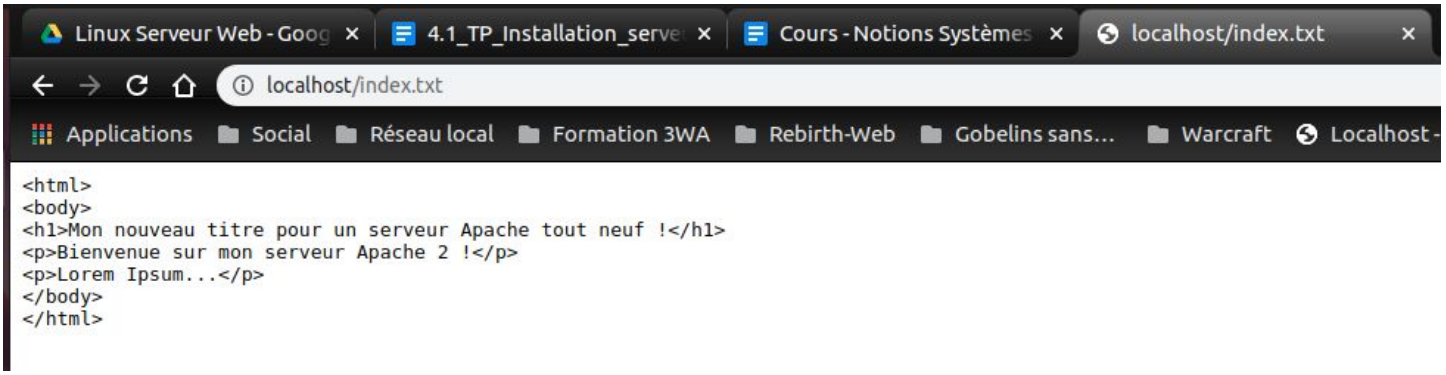
```
# cd /var/www
# mv index.html index.txt
```

Observons le résultat quand on se connecte simplement sur le site :

Nous voyons que ce n'est plus notre page web qui nous est présentée, mais le contenu du répertoire /var/www/html.

Si nous cliquons maintenant sur index.txt, que se passe-t-il ?

Ce n'est plus notre belle page web, mais simplement son contenu avec les balises...



Mais que se passe-t-il ? Notre serveur ne fonctionne plus ?

1.2.2. Ajouter des fichiers d'index à Apache2

Si, il continue de très bien fonctionner, mais il fait ce qu'on lui dit de faire.

Dans un premier temps, nous sommes allés vers la page principale du site sans préciser de nom de page spécifique. Dans ce cas, Apache regarde s'il y a un fichier par défaut qui doit s'appeler index et une extension d'un langage web.

Cette directive, très importante, se trouve dans le module dir.

```
# cat /etc/apache2/mods-available/dir.conf
<IfModule mod_dir.c>
        DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm
</IfModule>
```

La directive DirectoryIndex indique à Apache quel peut être le nom du fichier qui sera automatiquement lu par le serveur si aucun fichier n'est indiqué.

Nous pouvons essayer d'ajouter notre fichier index.txt au DirectoryIndex.

```
DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm index.txt
```

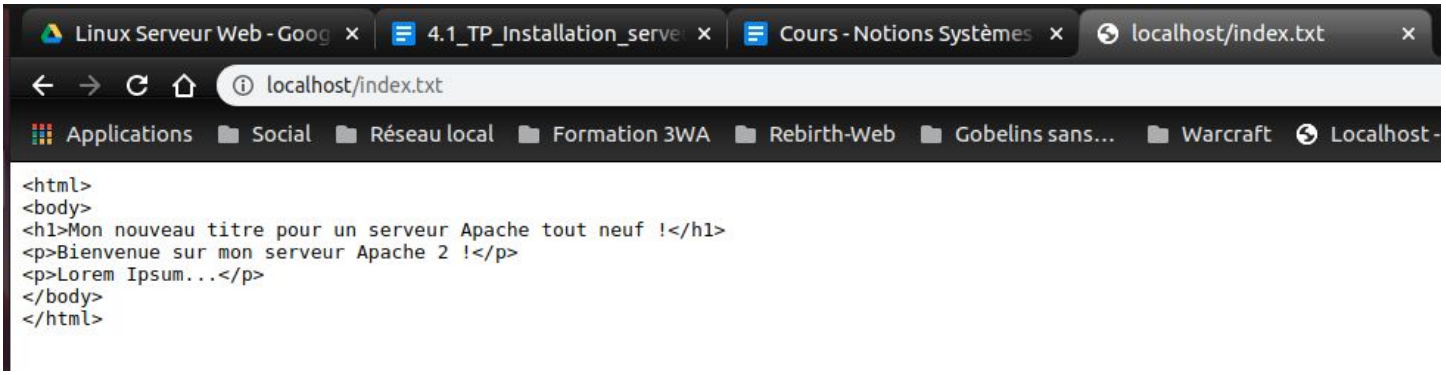
Regardons le résultat, en ne rentrant dans l'URL QUE l'adresse IP du site 192.168.0.1 ou localhost :

Eh bien ? Ça n'a rien changé...

Oui, car nous avons changé la configuration d'Apache2, mais nous ne lui avons pas dit. En fait, il faut relancer le service apache2 pour que les nouveautés de configuration soient prises en compte :

```
root@Predator:/var/www/html# service apache2 reload
```

Et nous pouvons maintenant rafraîchir la page :



On voit effectivement qu'Apache2 ne nous affiche plus le contenu du répertoire /var/www/, mais directement le fichier index.txt qui est bien spécifié dans le DirectoryIndex.

Cependant, c'est toujours le texte avec les balises qui est affiché, et non la page web interprétée.

1.2.3. Ajouter des types à Apache2

Pour cela, nous devons dire à Apache2 que les fichiers d'extension .txt doivent être interprétés comme des fichiers html. Cela se fait grâce à la directive AddType qui se trouve dans le module mime.

On ajoute à la fin du fichier :

```
AddType text/html .txt
```

On relance Apache et on actualise notre page, en appuyant sur la touche CTRL+F5 pour forcer à recharger la page :

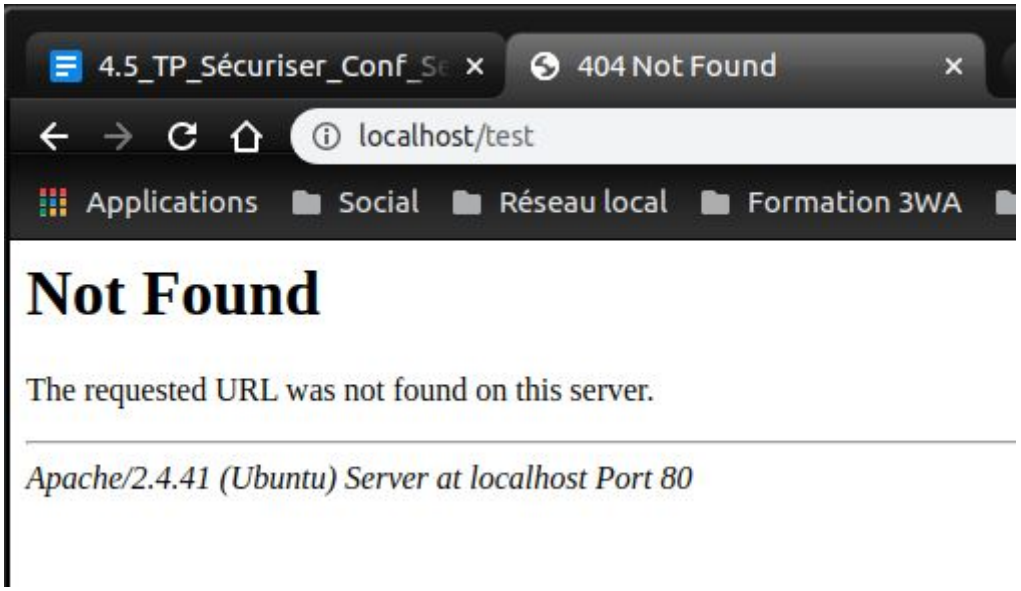
Cette fois tout est bon !

Nous avons vu quelques directives intéressantes d'Apache2, nous verrons un peu plus tard comment héberger plusieurs sites différents sur notre serveur grâce aux virtualhost.

1.3. Rendre Apache moins bavard

Par défaut, Apache est un peu trop bavard… On peut s'en rendre compte en accédant à notre serveur dans un navigateur. On tombe sur la page par défaut.

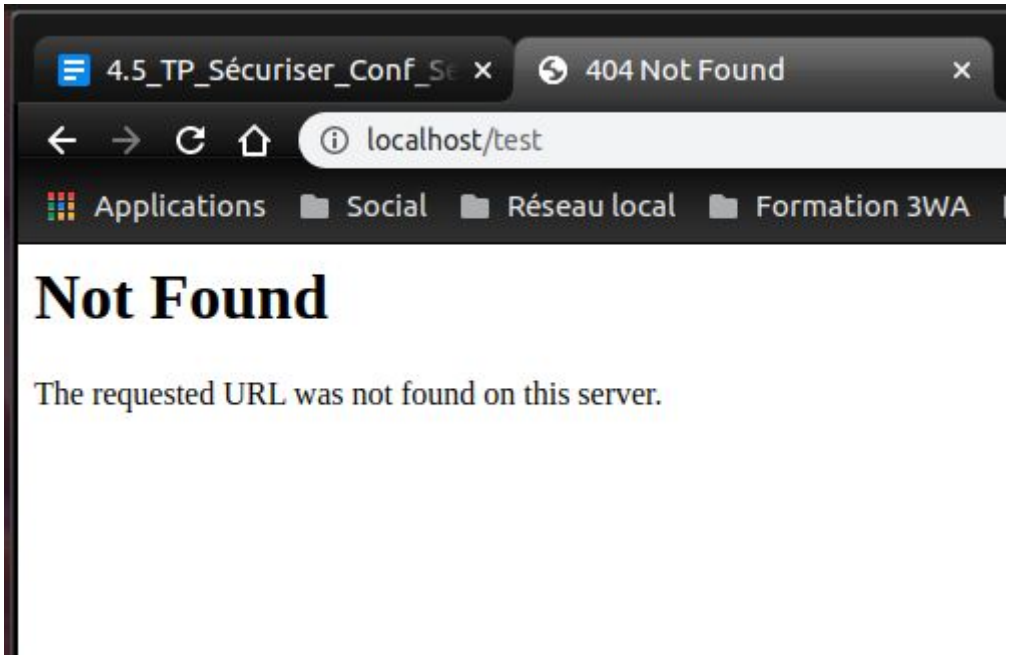
Essayons maintenant d'accéder à la page http://localhost/test on se rend vite compte qu'Apache indique sa version, l'OS ainsi que son port en bas de page. Des informations qui peuvent s'avérer utiles à toute personne souhaitant obtenir des informations sur une future cible potentielle.



Pour corriger cela, on va se rendre dans le fichier de configuration security.conf du répertoire conf-available et modifier les valeurs suivantes :

```
ServerTokens Prod
ServerSignature Off
```

Redémarrez le service Apache2 puis afficher à nouveau la page.



1.4. Restriction sur les répertoires

On va à présent éditer /etc/apache2/sites-available/000-default.conf afin de le comprendre et gérer plus finement les permissions sur les fichiers et répertoires.

```
# Securisation
<Directory /var/www/> # Racine d'Apache
     Order Deny,Allow
     Deny from all #On interdit l'accès à la racine de notre serveur à Apache
     Options None # On désactive toutes les options
     AllowOverride None #On interdit tous les fichiers .htaccess
     Order Allow,Deny
     #Allow from all
     Options -Indexes -ExecCGI +FollowSymlinks MultiViews # On empêche l'exécution de script
CGI ainsi que le parcours des répertoires
</Directory>
```

1.5. Vérifier la configuration d'Apache2 avant de redémarrer

Lors d'une modification de la configuration d'Apache, nous sommes bien souvent obligé de redémarrer le service "apache2" présent sur notre serveur. Le problème est que si la configuration n'est pas correcte, le service peut refuser de redémarrer. Si c'est sur un serveur en production, les sites hébergés seront indisponibles le temps de la réparation du problème.

Dans ce cas précis, il est serait plus judicieux de savoir vérifier si la configuration d'Apache2 est correcte avant de procéder à un redémarrage.

Il existe pour cela une commande fournis avec les paquets "apache2" qui se nomme "apachectl". Cette commande, dans notre cas, peut prendre l'option "configtest" qui va procéder à une vérification de la syntaxe des fichiers de configuration :

```
apachectl configtest
```

Si il y a un élément dans les fichiers de configuration qu'Apache2 ne comprends pas, il vous l'indiquera à la suite de la commande et cela vous permettra de le corriger sans avoir une indisponibilité du service et des sites qui s'y situent.

1.6. Lister les modules chargés par Apache

Pour cela, on va utiliser la commande « apachectl » qui permet de contrôler le daemon Apache comme par exemple le démarrer, l'arrêter, etc... Et évidemment lister les modules Apache chargés.

Afin de générer la liste des modules chargés, saisissez la commande suivante :

```
apachectl -t -D DUMP_MODULES
```

Exemple de résultat :

```
Loaded Modules:
 core_module (static)
 so_module (static)
 watchdog_module (static)
 http_module (static)
 log_config_module (static)
 logio_module (static)
 version_module (static)
 unixd_module (static)
 access_compat_module (shared)
 alias_module (shared)
 auth_basic_module (shared)
 authn_core_module (shared)
 authn_file_module (shared)
 authz_core_module (shared)
 authz_host_module (shared)
 authz_user_module (shared)
 autoindex_module (shared)
 deflate_module (shared)
```

```
dir_module (shared)
env_module (shared)
evasive20_module (shared)
filter_module (shared)
headers_module (shared)
mime_module (shared)
mpm_prefork_module (shared)
negotiation_module (shared)
php7_module (shared)
reqtimeout_module (shared)
security2_module (shared)
setenvif_module (shared)
status_module (shared)
unique_id_module (shared)
```

1.7. Désactivation de la méthode TRACE sous Apache

La méthode **HTTP TRACE** est une méthode souvent utilisée pour le débogage ou la recherche d'information. Celle-ci permet en effet de retourner au client exactement ce que le serveur à reçu. On peut alors détecter des changements d'information ou des informations permettant de comprendre les éventuelles modifications faites sur les **headers** par des proxy par exemple. Lorsque l'on fait un scan de site avec des outils comme **Nitko,** il est fréquent que celui-ci remonte une alerte concernant l'existence de la prise en compte de la méthode HTTP TRACE qui présente une vulnérabilité de type **XST** (Cross-Site-Tracing).

C'est pourquoi nous allons ici voir comment désactiver la prise en compte de la méthode HTTP TRACE. Les **attaques XST** utilisent la présence de la prise en compte de la méthode **HTTP TRACE** et d'une faille de type **XSS** pour que le pirate récupère le retour de la méthode TRACE par exemple. Pour exemple, voici le retour d'une commande TRACE en ligne de commande, il est également fréquent que l'on envoie le contenu des Cookies aux sites web, ce contenu est également renvoyé par la méthode TRACE :

```
root@VLAD-0:/etc/apache2# curl -X TRACE http://127.0.0.1 -H "Cookie: SessionID=D
Jddzjdzijjdz H7ZDd9ZJ38"
TRACE / HTTP/1.1
User-Agent: curl/7.26.0
Host: 127.0.0.1
Accept: */*
Cookie: SessionID=DJddzjdzijjdzH7ZDd9ZJ38
```

Au travers une attaque XST, le pirate va donc chercher à récupérer les informations envoyées au serveur par le client comme éventuellement les Cookies.

La meilleure façon de se protéger de ce genre d'attaque est bien sûr de refuser tout simplement de répondre aux méthodes HTTP TRACE. Apache inclus cette possibilité, il faut pour cela s'assurer que la ligne suivante est bien présente dans "/etc/apache2/conf-available/security" :

```
TraceEnable Off
```

1.8. Aller plus loin

- Filtrer les accès par adresse IP
- Autoriser l'accès depuis une adresse IP avec require ip
- Déplacer l'emplacement des sessions
- Changer l'emplacement des sites par défaut

1.9. Utilisation des virtualhosts

Nous allons essayer de présenter deux sites différents sur notre serveur. Mais comment Apache saura différencier ces deux sites et saura lequel présenter lors d'une requête sur le port 80 ?

En fait, il y a plusieurs façons de faire des virtualhosts, mais nous allons nous baser sur la plus répandue en utilisant des noms de domaine différents pour nos sites.

Nous allons créer les sites monsite1.fr et monsite2.fr.

Ainsi, quand une requête arrivera, Apache2 pourra savoir si la demande est pour monsite1.fr ou monsite2.fr.

1.9.1. Configuration du DNS

Sur Internet, vous devez acheter un nom de domaine pour qu'il devienne accessible.

Dans notre cas, nous allons plutôt utiliser une petite astuce qui pourra vous être utile pour beaucoup d'autres choses. Nous allons utiliser une fonctionnalité qui permet de court-circuiter le fonctionnement normal du DNS grâce au fichier hosts.

Le fichier hosts est un fichier présent sur tous les systèmes, qui permet d'indiquer des associations entre nom de machine et adresse IP qui seront prioritaires par rapport au DNS.

Par exemple, si j'écris:

```
192.168.0.1   www.google.fr
```

La prochaine fois que j'essaierai d'aller vers www.google.fr, ma machine pensera que le serveur de Google se trouve à l'adresse 192.168.0.1 et enverra la requête à cette adresse.

Nous allons donc modifier notre fichier hosts pour y ajouter des associations pour www.monsite1.fr et www.monsite2.fr.

Le fichier hosts se trouve dans /etc.

Nous allons éditer le fichier hosts du poste sur lequel nous souhaitons consulter les sites et ajouter deux lignes en haut du fichier :

```
192.168.0.1   www.monsite1.fr
192.168.0.1   www.monsite2.fr
```

Désormais, dès lors que notre machine voudra accéder à un de ces deux sites, elle utilisera la résolution locale au lieu de passer par le DNS.

Nous pouvons tester en entrant www.monsite1.fr dans l'URL de notre navigateur :

```
Site www.monsite1.fr
Site www.monsite1.fr
```

Ça marche ! Par contre, pour l'instant, le serveur Apache nous présente deux fois la même page.

1.9.2. Configuration des virtualhosts.

Les virtualhosts se configurent dans /etc/apache2/sites-available. Nous allons donc créer deux nouveaux fichiers pour nos deux virtualhosts. Pour cela, nous allons simplement copier le fichier 000-default.conf, et modifier le contenu des fichiers copiés.

```
cd /etc/apache2/sites-available/
cp 000-default.conf monsite1.fr.conf
cp 000-default.conf monsite2.fr.conf
```

Ensuite, nous allons simplement modifier trois choses et ajouter :

- une directive ServerName pour indiquer le nom de notre virtualhost
- le DocumentRoot qui précise où se situent nos pages
- la balise Directory pour y indiquer notre nouveau chemin.

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        ServerName www.monsite1.fr

        DocumentRoot /var/www/monsite1.fr/
        <Directory />
                Options FollowSymLinks
                AllowOverride all
        </Directory>
        <Directory /var/www/monsite1.fr/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride all
                Order allow,deny
                allow from all
        </Directory>

        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
        <Directory "/usr/lib/cgi-bin">
                AllowOverride None
                Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
                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

    Alias /doc/ "/usr/share/doc/"
    <Directory "/usr/share/doc/">
        Options Indexes MultiViews FollowSymLinks
        AllowOverride None
        Order deny,allow
        Deny from all
```

```
        Allow from 127.0.0.0/255.0.0.0 ::1/128
    </Directory>

</VirtualHost>
```

Nous faisons de même pour monsite2.fr.

Il nous reste encore à activer ces virtualhosts dans sites-enabled/.

```
a2ensite monsite1.fr.conf
a2ensite monsite2.fr.conf
```

Nos virtualhosts sont prêts, il ne nous reste plus qu'à créer les répertoires et pages de nos sites :

```
mkdir /var/www/monsite1.fr/
touch /var/www/monsite1.fr/index.html
echo "<html><body><h1>Bienvenu sur monsite1.fr !</h1></body></html>" >
/var/www/monsite1.fr/index.html

mkdir /var/www/monsite2.fr/
touch /var/www/monsite2.fr/index.html
echo "<html><body><h1>Bienvenu sur monsite2.fr !</h1></body></html>" >
/var/www/monsite2.fr/index.html
```

Nos virtualhosts devraient maintenant être effectifs.

1.10. Sécurisation du répertoire root d'un site

Pour attribuer les droits correctement à un répertoire root d'un site hébergé, il faut au préalable avoir créer un utilisateur et un groupe correspondant à votre site.

```
cd /var/www/nomdusite
chown www-monsite:www-monsite -R *
find . -type f -exec chmod 644 {} \;
find . -type d -exec chmod 755 {} \;
```