CONFIGURATION AVANCEE D'APACHE HTTP SERVER
Sécurisation des échanges client-serveur
Les fichiers `.htaccess` :

Objectif :

Utiliser des fichiers `.htaccess` afin de sécuriser l'accès à certaines ressources du serveur. Utiliser les directives Apache.

Prérequis :

Copiez un fichier index.html en le renommant 401.html et placez-le à la racine de votre serveur Web. Ce fichier (après avoir été modifié pour faire apparaître « Authentification obligatoire ») sera utilisé pendant le TP dans divers répertoires.

Copiez un fichier index.html en le renommant 403.html et placez-le à la racine de votre serveur Web. Ce fichier (après avoir été modifié pour faire apparaître « Accès refusé ») sera utilisé pendant le TP dans divers répertoires.

Copiez un fichier index.html en le renommant 404.html et placez-le à la racine de votre serveur Web. Ce fichier (après avoir été modifié pour faire apparaître « Page WEB inexistante ») sera utilisé pendant le TP dans divers répertoires.

Activez le module `authz_groupfile` pour permettre l'utilisation de la directive `authgroupfile`.

Insérez la directive `AllowOverride` dans la directive `<Directory>` de votre site BTS1 :

```
<VirtualHost * :80>
…
…
<Directory /var/www/html/bts1/>
AllowOverride All
</Directory>
…
…
</VirtualHost>
```

Sur votre machine cliente, vous utiliserez le navigateur Chrome, il serait intéressant de remarquer les différences de traitement avec Mozilla Firefox et/ou Microsoft Edge.

Votre machine cliente sera amenée à changer d'IP pendant le déroulement du TP, vous pouvez également prévoir plusieurs clients différents.

Vous utiliserez des fichiers `.htaccess` dans votre hôte virtuel bts1, assurez-vous de son fonctionnement avant le début du TP. N'hésitez pas à réaliser un snapshot…

Rappel de cours :

- Les fichiers `.htaccess` sont des fichiers de configuration d'Apache, permettant de définir des règles dans un répertoire et dans tous ses sous répertoires (qui n'ont pas de tel fichier à l'intérieur)

- On peut les utiliser pour protéger un répertoire par mot de passe, ou pour changer le nom ou l'extension de la page index, ou encore pour interdire l'accès au répertoire.

- Les fichiers `.htaccess` peuvent être utilisés dans n'importe quel répertoire virtuel ou sous répertoire. Les principales raisons d'utilisation des fichiers `.htaccess` sont :
  - Gérer l'accès à certains fichiers
  - Ajouter un Mime-Type
  - Protéger l'accès à un répertoire par un mot de passe
  - Protéger l'accès à un fichier par un mot de passe
  - Définir des pages d'erreurs personnalisées

- Le fichier `.htaccess` est placé dans le répertoire dans lequel il doit agir. Il agit ainsi sur les permissions du répertoire qui le contient et de tous ses sous répertoires.

- Vous pouvez placer un autre fichier `.htaccess` dans un sous répertoire d'un répertoire déjà contrôlé par un fichier `.htaccess`. Le fichier `.htaccess` du répertoire parent reste en « activité » tant que les fonctionnalités n'ont pas été réécrites.

- Pensez à bien positionner la directive `AllowOverride` pour que vos fichiers `.htaccess` soient pris en compte

- Un fichier `.htaccess` est composé de deux sections :

- Une première section contient les chemins des fichiers contenant les définitions de groupes et d'utilisateurs :
  AuthUserFile /repertoire/du/fichier/.FichierMotDePasse
  AuthGroupFile /repertoire/du/fichier/.FichierGroupe
  AuthName « Accès protégé »
  AuthType Basic
  - `AuthUserFile` définit le chemin d'accès absolu vers le fichier de mot de passe
  - `AuthGroupFile` définit le chemin d'accès absolu vers le fichier de groupe
  - `AuthName` entraîne l'affichage dans le navigateur Internet de : « Tapez votre nom d'utilisateur et votre mot de passe. Domaine : « Accès protégé »
  - `AuthType Basic` précise qu'il faut utiliser `AuthUserFile` pour l'authentification

- Une seconde section contient la définition des conditions d'accès :
  `Require valid-user {instruction d'accès à satisfaire}`
  - `Require valid-user` précise que l'on autorise uniquement les personnes identifiées. Il est également possible de préciser explicitement le nom des personnes autorisées à s'identifier : `Require user {username}`

Travail à réaliser :

1. Protéger un répertoire par un mot de passe

Il s'agit d'une des applications les plus utiles du fichier `.htaccess` car elle permet de définir de façon sûre (à l'aide d'un login et d'un mot de passe) les droits d'accès à des fichiers par certains utilisateurs. La syntaxe est la suivante :

```
AuthUserFile {emplacement du fichier de mot de passe}
AuthGroupFile {emplacement du fichier de groupe}
AuthName "Accès protégé"
AuthType Basic
Require valid-user
```

- La directive `AuthUserFile` permet de définir l'emplacement du fichier contenant les logins et les mots de passe des utilisateurs autorisés à accéder à une ressource donnée.

- La directive `AuthGroupFile` permet de définir l'emplacement du fichier contenant les groupes d'utilisateurs autorisés à s'identifier. Il est possible d'outrepasser cette déclaration en déclarant le fichier suivant : `/dev/null`.

Voici un exemple de fichier `.htaccess` :

```
ErrorDocument 401 /401.html
AuthUserFile /var/pass/.file
AuthGroupFile /dev/null
AuthName "Accès sécurisé au site BTS 1"
AuthType Basic
Require valid-user
```

Créez un fichier `.htaccess` qui demande de s'identifier pour accéder au répertoire secret que vous avez créé. Utilisez un fichier de mot de passes contenant au moins 4 noms et vérifiez que seulement ceux-ci fonctionnent.

Le fichier de mot de passe est un fichier texte devant contenir sur chacune de ses lignes le nom de chaque utilisateur suivi des deux points (:), puis du mot de passe crypté (solution recommandée) ou en clair. Ce fichier de mot de passe ne devrait pas être mis dans un répertoire virtuel Internet.

Voici un exemple de fichier de mots de passe non chiffrés (ici `.file`). ***Ne pas utiliser ce fichier mais utiliser la version chiffrée*.** Les versions non chiffrées ne sont plus supportées par défaut par les navigateurs actuels.

```
utilisateur:password
eleve:motdepasse
```

Voici le même fichier contenant des mots de passe chiffrés :

```
utilisateur:AOe7eG7vzh
eleve:UMO9AnrNfM
```

Apache fournit un outil permettant de générer facilement des mots de passe cryptés, il s'agit de l'utilitaire `htpasswd` accessible dans le sous-répertoire `bin` d'Apache. La syntaxe de cet utilitaire est la suivante :

Pour créer un nouveau fichier de mots de passe :

```
htpasswd -c {chemin du fichier de mot de passe} utilisateur
```

Pour ajouter un nouvel utilisateur/mot de passe à un fichier existant

```
htpasswd {chemin du fichier de mot de passe} utilisateur
```

2. Empêcher l'accès à un répertoire par un domaine

La syntaxe pour bloquer l'accès d'un répertoire par un domaine est la suivante :

```
Order (allow, deny ou deny,allow)
Allow (all, [liste de domaine])
Deny (all, [liste de domaine])
```

Exemple :

```
Order deny,allow
Deny from .domaine.com
```

Toutes les requêtes provenant du domaine `.domaine.com` ne pourront avoir accès aux ressources comprises dans le répertoire et ses sous-répertoires. La commande `Order` sert à préciser explicitement que la commande `Deny` va bien annuler l'effet de `Allow` et non l'inverse.

Ici une restriction d'accès plus complète :

```
ErrorDocument 401 /401.html
ErrorDocument 403 /403.html
AuthUserFile /var/pass/.file
AuthGroupFile /dev/null
AuthName "Accès sécurisé au site BTS 1"
AuthType Basic
order deny,allow
deny from all
allow from 192.168.0.12
require user utilisateur eleve
```

L'accès ne sera possible que pour les utilisateurs « utilisateur » et « eleve » à partir de l'adresse IP 192.168.0.12 et avec le bon mot de passe.

Faites le nécessaire pour accéder correctement au site `bts1` avec cette restriction d'accès.

Modifier le fichier `.htaccess` pour retirer la restriction par IP.

3. Empêcher l'accès à un fichier particulier

Par défaut, Apache applique les restrictions du fichier `.htaccess` à l'ensemble des fichiers du répertoire dans lequel il se trouve ainsi qu'à tous les fichiers contenus dans ses sous-répertoires. Il est également possible de restreindre l'accès pour un ou plusieurs fichiers du répertoire grâce à la balise *<Files>.*

Voici un exemple de restriction aux fichiers *secret.html* et *secret2.html* :

```
<Files secret.html>
AuthUserFile /var/pass/.file
AuthGroupFile /dev/null
AuthName "Accès sécurisé au site BTS 1"
AuthType Basic
require user eleve
</Files>
```

```
<Files secret2.html>
AuthUserFile /var/pass/.file
AuthGroupFile /dev/null
AuthName "Accès sécurisé au site BTS 1"
AuthType Basic
require valid-user
</Files>
```

Créez un fichier `.htaccess` qui demande de s'identifier pour accéder à l'url http://www.bts1.sio/secret/info.htm et qui limite cet accès à l'utilisateur igor. Il ne faut utiliser qu'une seule balise `<Files>` par fichier. Sinon, l'erreur suivante est reportée dans le fichier de log des erreurs :

```
.htaccess: Multiple arguments not (yet) supported.
```

Pour info, depuis Apache 1.3, pour traiter plusieurs fichiers, il est conseillé d'utiliser la balise `<FilesMatch>` à la place de la balise `<Files>`. Cette nouvelle balise ne supporte aussi qu'un seul argument mais on peut traiter plusieurs fichiers grâce à une expression régulière.

4. Empêcher l'accès à un type de fichiers par un réseau

```
<Files *.png>
Deny from 192.168.0.0
</Files>
```

Toutes les requêtes provenant du réseau 192.168.0.0 ne pourront avoir accès aux images, dont l'extension est .png, comprises dans le répertoire et ses sous-répertoires. Créez un fichier `.htaccess` qui bloque la lecture des fichiers *.gif et *.txt dans le répertoire secret pour tous les utilisateurs. Vous pouvez créer un fichier .txt à l'aide de *nano*, vous pouvez récupérer un .gif depuis internet avec la commande *`wget`*, il faudra cependant que votre machine communique avec l'extérieur ou disposer d'une connexion `ssh` avec le serveur.

5. Protéger un répertoire par un login

Cette méthode (beaucoup moins sûre que la précédente) permet une authentification de bas niveau uniquement par le nom de l'utilisateur. La syntaxe est la suivante :

```
Require (user [liste des utilisateurs], group [liste des groupes], valid-user)
```

Voici un exemple de ligne du fichier `.htaccess` :

```
Require user eleve student arthur louis
```
Les utilisateurs acceptés sont : eleve, student, arthur, louis

```
Require group eleve
```
Les utilisateurs acceptés font partie du groupe eleve

```
Require valid-user
```
Les utilisateurs acceptés sont tous les utilisateurs authentifiés

Attention ici user est un mot clef et non le nom d'un utilisateur !!! Tout utilisateur souhaitant rentrer dans le répertoire ou un de ses sous-répertoires sera refusé sauf s'il s'identifie en donnant un nom figurant dans la liste.

6. Obliger un utilisateur à satisfaire à au moins une des conditions

Voici la syntaxe :

```
Satisfy (any, all)
```

```
Order deny,allow
Deny from all
Allow from 192.168.0.100
Require user eleve student arthur louis
Satisfy Any
```

Ce qui signifie que l'accès au répertoire sera bloqué pour tout le monde à l'exception des personnes qui s'identifient (eleve, student, arthur ou louis) ou des requêtes provenant de l'IP 192.168.0.100.

7. Personnalisation des messages d'erreur

Il s'agit d'une fonctionnalité pratique car elle permet de définir une page par défaut pour un type d'erreur donné. Cela permet d'une part de guider l'utilisateur au lieu d'afficher la banale page d'erreur du navigateur, ainsi que d'égayer la navigation même en cas d'erreur.

```
ErrorDocument (code-à-3-chiffres [nom du fichier ou texte ou url])
```

```
ErrorDocument 401 « Vous n'êtes pas enregistré »
```

```
ErrorDocument 404 /404.html
```

Chemin relatif depuis la racine de votre serveur web

```
ErrorDocument 403 http://bts.sio/403.html
```

Les deux lignes suivantes permettent de définir des pages d'erreurs personnalisées au cas où l'accès à un document serait interdit ou bien que le document n'existe pas :

```
ErrorDocument 403 /403.html
ErrorDocument 404 /404.html
```

Ceci vous permet de donner un message d'erreur personnalisé remplaçant les fichiers fournis avec le navigateur. Voici quelques-unes des erreurs les plus courantes à personnaliser :

```
401 Unauthorized : la personne n'a pas passé avec succès
l'identification.
403 Forbidden : le serveur n'a pas le droit de répondre à votre
requête.
404 Not Found : le serveur n'a pas trouvé le document souhaité.
```

8. Changer le fichier index par défaut

Le fichier `index.html` est le fichier qui est affiché lorsqu'aucun nom de fichier n'est défini dans l'URL. Cela permet d'éviter que le navigateur liste l'ensemble des fichiers contenus dans le répertoire (pour des raisons de confidentialité). On a déjà vu que la syntaxe pour effectuer ce type d'opération est la suivante :

```
DirectoryIndex (fichiers)
```

Voici un exemple de mise en application :

```
DirectoryIndex index.php index.html index.phtml /403.html
```

Lorsque vous essayez d'accéder au répertoire sans préciser la page à afficher, Apache va avoir recours à la directive `DirectoryIndex`. En général, par défaut, cette directive pointe vers `index.html` puis `index.htm`. Dans l'exemple ci- dessus, Apache va commencer par chercher `index.php`, puis `index.html`, et ensuite `index.phtml`. Si aucun de ces trois fichiers n'existent, la page `403.html` (se trouvant dans la racine) sera affichée pour éviter de lister le répertoire.