

Sommaire

Protection d'un serveur Apache PHP MySQL.....	1
1 Objectif de ce document.....	1
2 Introduction.....	1
3 Protection du serveur Apache.....	1
3.1 Protection par .htaccess.....	2
3.2 Protection par modification de httpd.conf.....	3
3.3 Différence entre les deux méthodes .htaccess et httpd.conf.....	4
3.4 Protection de fichiers particuliers dans un sous répertoire.....	4
3.5 Rendre le contenu d'un sous-répertoire invisible.....	5
4 Protection du gestionnaire de bases de données MySQL.....	5
4.1 Configuration des fichiers d'autorisations d'accès.....	6
4.2 Automatisation des connexions.....	9
5 Contrôle des accès MySQL initiés par des scripts PHP.....	10
6 Cas particulier de phpMyAdmin.....	10
6.1 Configuration avec adv_auth = false.....	11
6.2 Configuration avec adv_auth=true.....	12
7 Le petit bréviaire.....	13
8 Conclusion.....	14
L'auteur.....	14
Copyright.....	15

Protection d'un serveur Apache PHP MySQL

Protection d'un serveur Apache PHP MySQL

Jean-Marc LICHTLE

Protection d'un serveur Apache PHP MySQL contre les visites non souhaitées

1 Objectif de ce document

L'objectif est de décrire comment protéger un serveur constitué de la trilogie Apache + PHP + MySQL des ``visites inopportunes". La description est relative à une installation LINUX, plus spécifiquement la version 8.0 de MANDRAKE.

Exemple de problème de sécurité posé : vous avez installé phpMyAdmin. Il est souhaitable d'interdire qu'un visiteur non autorisé accède à ce script ce qui lui donnerait des droits dangereux sur toutes les bases de données installées.

Il n'entre pas dans les vues de l'auteur de transformer le lecteur en spécialiste de la sécurité informatique. Ce document veut être :

- Un pied à l'étrier pour tout utilisateur qui voudrait étudier la question de la sécurité d'un serveur Web Apache + PHP + MySQL.
- La description du minimum de précautions à mettre en place sur un serveur d'entreprise ne contenant aucune information stratégique mais qui mettrait à la disposition des utilisateurs des informations relativement banalisées. L'objectif de sécurité est relativement sommaire :
 - ◆ – éviter que des manipulations non souhaitées ne détruisent une partie ou la totalité des informations,
 - ◆ – limiter les droits d'accès à certaines informations plus sensibles à un nombre limité d'utilisateurs identifiés.

2 Introduction

Ce texte est la suite naturelle d'un document décrivant l'installation de la trilogie Apache PHP MySQL en environnement LINUX MANDRAKE 8.0 (voir l'article). L'installation décrite dans ce premier document est la plus simple possible, son seul objectif était d'aboutir à un système qui soit fonctionnel. Dans l'installation de MySQL par exemple j'avais précisé de faire l'impasse sur la mise en place d'un mot de passe administrateur afin de ne pas bloquer le fonctionnement ultérieur de phpMyAdmin dans son installation de base. Il est temps maintenant de faire le pas supplémentaire qui verrouillera un peu mieux le site.

Pour plus d'informations on pourra se reporter à l'importante littérature qui existe sur la question, notamment ``Pratique de MySQL et PHP", édition O'REILLY.

Il importe de comprendre que la protection d'un site Apache + PHP + MySQL prend des facettes multiples, chaque logiciel ayant ses ``protections" propres. De plus comme nous le verrons bientôt avec phpMyAdmin, ces systèmes de protections dialoguent entre eux.

3 Protection du serveur Apache

Il s'agit dans un premier temps de protéger le serveur http, c'est à dire le logiciel qui va adresser les pages html au navigateur qui en fait la demande. A ce stade il importe peu que PHP et MySQL soient exploités ou non. Cette protection est très générale et s'applique aussi bien à un serveur simple ne mettant en oeuvre que des pages html pures qu'à un serveur plus évolué faisant appel à un langage de programmation et à un gestionnaire de bases de données (PHP et MySQL ou une autre combinaison). Les exemples qui suivent portent sur la protection d'un sous répertoire `test/` situé au premier niveau de l'arborescence du serveur, plus précisément `/var/www/html/test/`. Attention, toutes les informations de chemins d'accès sont relatives à la distribution Mdk 8.0 !

3.1 Protection par .htaccess

Le principe consiste à créer dans le sous répertoire à protéger un fichier nommé `.htaccess` (commençant par un point donc caché) et qui va en limiter les droits d'accès. En fait `.htaccess` va renvoyer vers un fichier contenant les logins et mots de passe des utilisateurs autorisés.

Ce fichier de mots de passe peut porter un nom quelconque. Habituellement ce dernier est rangé dans `/etc/httpd/auth/`, répertoire qu'il convient éventuellement de créer puisqu'il n'existe pas après l'installation de base. Il est assez logique que tous les fichiers d'authentification seront rangés dans le même sous-répertoire. Il est de ce fait judicieux de choisir des noms de fichiers qui soient parlant et qui rappellent l'objet de la protection, par exemple `test.users` pour le fichier qui définira les droits d'accès des différents utilisateurs au sous répertoire `test`. L'emploi de `/etc/httpd/auth/` pour ranger les fichiers d'autorisation est une pure question de convention. Ces fichiers pourraient aussi bien être rangés n'importe où ailleurs. Il va toutefois sans dire (mais mieux en le disant) que ces fichiers doivent être hors de portée des visiteurs, en clair ne doivent pas faire partie de l'arborescence `/var/www/html/...`

3.1.1 Création de .htaccess

Exemple simple :

Avec un éditeur quelconque (de préférence emacs =;-) créer le fichier minimum suivant:

```
AuthUserFile /etc/httpd/auth/test.users

AuthName "Accès restreint"

AuthType Basic

require valid-user
```

Sauvegardez...

`AuthUserFile` définit quel est le fichier chargé de contenir les autorisations. La Mandrake 8.0 suppose par défaut que les fichiers d'autorisation sont rangés dans `/etc/httpd/`. La première partie de l'adresse `AuthUserFile` peut donc être omise éventuellement.

`AuthName` précise le message qui sera affiché dans la boîte de dialogue, dans ce cas ``Accès restreint à localhost" si la machine est nommée localhost. Nous verrons plus loin l'intérêt de choisir un message aussi explicite que possible.

3.1.2 Création du fichier des autorisations d'accès

La première étape consiste à créer, s'il n'existe pas encore, le sous répertoire `/etc/httpd/auth/` qui contiendra les autorisations (md... sous compte root). Le fichier d'autorisation lui-même se crée (et s'entretient) avec la commande `htpasswd`. Attention: la commande `htpasswd` peut être lancée aussi bien par un utilisateur de base que par l'administrateur. Toutefois, les fichiers contenus dans `/etc` étant tous la propriété de root je préconise, dans un but de cohérence, de lancer cette commande uniquement sous compte root. De toute façon, sauf à imaginer que `/etc/httpd/auth/` soit propriété d'un utilisateur non root, le lancement de `htpasswd` depuis un login non root conduirait à un échec à l'enregistrement des données.

Syntaxes:

- Pour la création (avec un premier utilisateur lambda): `htpasswd -c /etc/httpd/auth/test.users lambda`. `htpasswd` va alors vous demander le mot de passe de lambda (avec confirmation).
- Pour l'ajout d'un utilisateur supplémentaire: `htpasswd /etc/httpd/auth/test.users lambda2`
- Pour supprimer un utilisateur et ben le plus simple est de supprimer la ligne correspondante dans le fichier d'autorisation !

La consultation de `.htaccess` se fait à chaque accès du serveur Apache au sous-répertoire concerné. Avantage : la nouvelle configuration entre en action immédiatement sans qu'il soit nécessaire de relancer le serveur Apache pour recharger la nouvelle configuration.

3.2 Protection par modification de `httpd.conf`

Le principe est assez similaire à ce qui a été exposé ci-dessus si ce n'est que le renvoi vers le fichier des autorisations est réalisé dans `/etc/httpd/conf/httpd.conf`.

Pour mettre en place ce renvoi éditez ce fichier et ajoutez (par exemple tout à la fin) les lignes suivantes :

```
<Directory /var/www/html/test>

AuthName "Accès limité"

AuthUserFile /etc/httpd/auth/test.users

AuthType Basic

require valid-user

</Directory>
```

Pour protéger plusieurs répertoires créez plusieurs paragraphes `<Directory>` `</Directory>`.

Sauvegardez puis relancez le démon Apache (comme root : `/etc/rc.d/init.d/httpd restart`). En effet dans ce cas vous venez de modifier un des fichiers de configuration du serveur, fichier qui est lu au moment du chargement de Apache. Il convient donc de forcer Apache à relire ce fichier pour prendre en compte les modifications.

3.3 Différence entre les deux méthodes .htaccess et httpd.conf

Votre attention aura certainement été attirée par la similitude de rédaction entre les deux solutions. Cette similitude n'est absolument pas fortuite et provient simplement de la logique interne du fonctionnement d'Apache. Le serveur lit au lancement les paramètres de configuration stockés dans httpd.conf. Il ne les relira ensuite qu'en cas de demande explicite (restart).

Par contre, il cherchera à chaque requête d'un navigateur, à lire un éventuel fichier .htaccess contenu dans le sous-répertoire concerné par la demande. S'il le trouve les informations contenues dans ce fichier remplaceront, pour la durée de la requête, celles qui sont contenues dans httpd.conf. En clair vous pouvez "masquer" la configuration officielle contenue dans httpd.conf en proposant une nouvelle configuration dans un .htaccess. Faites l'essai en mettant en place une protection double :

- Par httpd.conf en mettant "Accès contrôlé par httpd.conf" dans la variable AuthName.
- Par .htaccess en mettant "Accès contrôlé par htaccess" dans AuthName.

Vous pourrez vérifier simplement (en renommant .htaccess par exemple) que la protection par .htaccess, lorsqu'elle est en place, remplace bien la protection induite par httpd.conf.

L'intérêt de cette petite démonstration ne semble pas évident. Trois éléments méritent toutefois d'être retenus :

- Une protection par .htaccess nécessite un travail supplémentaire d'analyse de la part du serveur (à prendre en compte pour les serveurs très chargés ou installés sur des machines poussives).
- Certaines directives de httpd.conf peuvent être neutralisées, modifiées...
- Une protection par .htaccess est prise en compte immédiatement sans nécessité d'intervenir sur le serveur.

3.4 Protection de fichiers particuliers dans un sous répertoire

La protection par .htaccess décrite ci-dessus s'applique à l'ensemble d'un sous-répertoire. En fait vous pouvez différencier, si vous le souhaitez, les accès aux fichiers. La syntaxe de .htaccess s'en trouve légèrement modifiée, par exemple pour protéger un fichier nommé `common.php` :

```
<Files common.php>

AuthName "Accès limité au fichier common.php par
.htaccess"

AuthUserFile /etc/httpd/auth/common.users

AuthType Basic

require valid-user

</Files>
```

L'intérêt de libeller AuthName de façon explicite apparaît ici de façon nette. Lors des essais vous pourrez en effet savoir immédiatement quels sont les contrôles d'accès qui sont en place.

Vous pouvez souhaiter protéger plus spécifiquement l'accès à un fichier particulier. La solution consiste à créer un fichier `.htaccess` libellé comme suit :

Exemple pour protéger un fichier `common.php` :

```
<Files common.php>

Order Deny,Allow

Deny from All

</Files>
```

Pour protéger plusieurs fichiers il suffit de créer plusieurs rubriques `<Files> </Files>`.

Attention: cette méthode interdit l'accès via le serveur Web pour tout le monde, y compris le propriétaire du fichier, l'administrateur, etc.

3.5 Rendre le contenu d'un sous-répertoire invisible

L'astuce qui suit consiste simplement à faire afficher par le serveur Web une page html qui signale au visiteur qu'il n'a rien à faire dans ce sous-répertoire.

Lorsque Apache accède à un sous-répertoire pour satisfaire à la demande d'un navigateur client il commence par vérifier les autorisations d'accès (voir ci-dessus). S'il ne rencontre pas de veto alors il part à la recherche d'un éventuel fichier `index.html` ou `index.php` (si PHP est installé). S'il ne trouve pas ce fichier alors il affiche le contenu du sous-répertoire qui apparaît donc en clair pour le visiteur.

Pour éviter que le contenu du sous-répertoire ne s'affiche il suffit donc de créer un fichier `index.html` ou `index.php`. Concevoir le contenu de ce fichier de telle sorte qu'il réponde à votre attente, du simple message d'avertissement au script PHP qui va renvoyer de force le visiteur dans le droit chemin. C'est simple et relativement efficace pour un environnement non critique. La solution la plus simple consiste simplement à créer un fichier `index.html` vide. Le navigateur recevra donc une page blanche. Pas très explicite mais simple et efficace !

Attention, il s'agit là simplement de créer une dissuasion. Nous ne sommes plus dans le domaine de la protection gérée comme ci-dessus. Un utilisateur qui connaîtrait un ou plusieurs noms de fichiers contenus dans ce sous-répertoire pourrait y accéder sans le moindre problème en tapant simplement leurs URL complets ! De grâce, ne construisez pas un site central de banque avec de telles méthodes !

4 Protection du gestionnaire de bases de données MySQL

Un peu comme dans le cas d'Apache cette section est spécifique à MySQL. Il importe peu pour ce qui va suivre qu'Apache soit installé ou non (idem pour PHP). Nous supposons que les notions de bases du langage SQL sont connues (ben voyons !). La première partie de ce chapitre va en effet faire largement appel à ce langage pour régler la configuration. Attention, nous allons utiliser le client texte MySQL ! Efficace mais pour le moins spartiate !

Pour commencer ouvrez une fenêtre texte (xterm, rxvt, terminal ou autre) et lancez le client MySQL par :

mysql -u root (on suppose que vous êtes logué comme root).

La réponse devrait être immédiate ``Welcome to the MySQL Monitor" et se terminer par l'invite de commande de MySQL qui est modestement mysql>. Je reproduirais cette invite pour tous les exemples de syntaxe SQL donnés ci-dessous.

Attention, vous êtes logué comme administrateur, tout ce que vous allez faire pourra devenir dramatique en cas d'erreur !

tapez mysql>use mysql ; pour préciser au serveur d'utiliser dans ce qui suit la base de données nommée mysql.

La réponse devrait se terminer par ``Database changed".

4.1 Configuration des fichiers d'autorisations d'accès

La protection de MySQL s'articule autour des fichiers d'autorisations (qui constituent eux-même une base de données MySQL) qui définissent très précisément les droits de chaque utilisateur, des informations les plus globales (telle base de donnée est accessible à tel utilisateur) aux plus pointues (la n-ième colonne de telle table est accessible pour tel utilisateur mais seulement pour telle opération).

Ces droits sont différenciés, consultation, mise à jour, etc.

La base de données qui décrit la structure des autorisations d'accès est contenue dans /var/lib/mysql/mysql/ (dans l'installation Mdk8.0). Cette BDD est créée automatiquement par le script d'installation du package MySQL. Vous pouvez consulter la liste des tables de cette base par mysql>show tables;

Cette commande affiche la liste des tables constituant la BDD.

L'objet du présent document étant de constituer une aide de départ (et non de remplacer la documentation spécialisée) nous ne nous intéresserons dans ce qui suit qu'à une seule table de cette base de données, la table user qui définit les droits d'accès globaux des utilisateurs à l'ensemble des BDD. De même nous n'étudierons que le cas d'une machine isolée servant à la fois de serveur et de client. La configuration en réseau n'est guère plus complexe mais dépasserait le cadre de la présente étude.

4.1.1 Structure de la table user

La structure de la table user est assez simple. Elle contient les champs suivants:

Host char(60)	nom de la machine depuis laquelle est fait l'appel
User char(16)	login de l'utilisateur
Password char(16)	mot de passe codé
Select_priv	droits d'effectuer des requêtes sélect (valeur Yes ou No)
Insert_priv	...
Update_priv	...
Delete_priv	...
Create_priv	droits d'effectuer une création de table

Drop_priv	droits de suppression d'une table
Reload_priv	droits de relancer le serveur MySQL
Shutdown_priv	droits d'arrêter le serveur MySQL
Process_priv	
File_priv	
Grant_priv	
References_priv	
Index_priv	
Alter_priv	

Cette structure peut être consultée très simplement par `mysql>desc user;` Son contenu peut être affiché par `mysql>select * from user;`

4.1.2 Mise à jour de la table user

Cette table contient d'origine (après l'installation décrite dans le document précédent) 4 enregistrements qui définissent les droits de l'administrateur depuis localhost et localhost.localdomain et ceux d'un utilisateur non nommé depuis les mêmes hôtes. L'administrateur root (dont vous utilisez le compte actuellement) a tous les droits, l'utilisateur non nommé aucun, si ce n'est celui d'accéder à la base de données mais sans pouvoir faire la moindre opération. Vous constaterez également que root n'a pas de mot de passe (voir plus haut, cette partie du script a été sautée au moment de l'installation de façon à ne pas bloquer le fonctionnement de phpMyAdmin).

La première étape va consister à supprimer les lignes qui sont inutiles de telle sorte à ne conserver qu'une seule ligne, celle qui correspond à l'administrateur root depuis localhost.

Tapez donc :

```
mysql>delete from user where Host='localhost.localdomain';
```

```
mysql>delete from user where User='';
```

```
mysql>select * from user;
```

Cette fois la liste devrait contenir le seul enregistrement relatif à root depuis localhost.

Pour ajouter un utilisateur lambda procédez comme suit :

```
mysql>insert into user
values('localhost','lambda',password('mdp_lambda'),
'Y','Y','Y','Y','N','N','N','N','N','N','N','N','N','N');
```

Ne comptez pas, il y a 4 'Y' et 10 'N' !

Vous venez de créer un nouvel utilisateur nommé lambda et dont le mot de passe est mdp_lambda (vous mettez bien sûr ce qui vous convient).

La syntaxe de cette ligne mérite quelques commentaires :

- Insert into user ... donne l'ordre de créer une nouvelle ligne dans la table user.
- Une ligne insert complète (avec toutes les options SQL) préciserait la liste des champs à compléter, dans ce cas particulier nous donnons exactement autant de valeurs (values) qu'il y a de champs dans la table, l'affectation des valeurs aux champs devient donc implicite (et la ligne plus simple !).
- La liste values() contient exactement 17 valeurs séparées par des virgules. Les chaînes de caractères sont placées entre guillemets simples (').
- La composition de cette liste reflète précisément la structure de la table user mise en évidence précédemment avec `mysql>desc user;`
- Le nouvel utilisateur dispose de 4 droits simples, sélection, insertion de nouvelles lignes, modifications de lignes existantes et effacement de lignes. Il ne peut ni créer une nouvelle table, ni en effacer une existante. Toutes les fonctions d'administration lui sont interdites.
- Password() est une fonction texte qui transforme un mot de passe donné en clair en une chaîne codée.

Vous pouvez maintenant quitter le client mysql en tapant `mysql>quit;`

4.1.3 Rechargement de la table des autorisations par MySQL

Il reste à vérifier la validité de ce nouvel utilisateur en essayant de se connecter avec ce nouveau login. Mais avant il faut que MySQL actualise la table des autorisations conservée en mémoire et chargée au lancement initial du serveur.

Pour cela tapez dans un terminal :

```
# mysqladmin flush-privileges
```

Cette commande demande à MySQL de recharger les tables définissant les autorisations d'accès depuis le disque sans pour autant arrêter le serveur.

Une autre solution plus brutale serait d'arrêter le serveur et de le redémarrer en tapant (compte root) `# /etc/rc.d/init.d/mysql stop` puis `# /etc/rc.d/init.d/mysql start`.

On peut enfin arrêter complètement LINUX et relancer mais là c'est la honte, n'est-ce pas ? Nous ne sommes pas sous Window\$ que diable !

4.1.4 Login avec le nouveau compte

Vous pouvez maintenant vous loguer avec le compte nouvellement créé. Plusieurs possibilités s'offrent à vous :

- `$ mysql -u lambda -pmdp_lambda`
Cette syntaxe vous donne l'accès direct. Attention: vous laissez un espace après `-u` mais pas après `-p`, collez directement le mot de passe à `-p` ! Inconvénient : le mot de passe apparaît en clair, sur l'écran mais aussi dans l'historique des frappes au clavier.
- `$ mysql -u lambda -p`
Avec cette syntaxe le serveur MySQL vous demandera immédiatement le mot de passe associé au login lambda. Vous devrez taper ce mot de passe en aveugle. Il n'apparaîtra donc ni à l'écran, ni dans un historique de frappes.
- `$ mysql -u lambda -p mysql`
Variante de la précédente, l'espace après `-p` va provoquer une demande de mot de passe comme ci-dessus, l'indication de mysql va indiquer au serveur d'utiliser la base de donner mysql (équivalent à `mysql>use mysql;`).

Ce qui ne marche pas:

- `$ mysql -u lambda`
MySQL va protester parce que l'utilisateur lambda est connu comme étant validé par un mot de passe. Le message sera assez explicite puisqu'il se termine par `Using password: YES`. Nota : voir plus loin le chapitre Automatisation des connexions, il sera possible d'automatiser l'indication du mot de passe et donc de se connecter en utilisant cette syntaxe.

Voilà, vous êtes maintenant en mesure de modifier efficacement le fichier user pour accorder et retirer des droits à chacun des utilisateurs de votre système. Ce chapitre n'a simplement fait que soulever un coin du voile sur la question. L'étude de l'utilisation des autres fichiers de `/var/lib/mysql/mysql/` vous permettra de définir encore mieux ces droits et de créer des droits différenciés par BDD et par utilisateur, voire même par table ou par champ. L'étape suivante s'articulerait autour de la table db. Cette étude de détail sortirait toutefois du cadre fixé ici.

4.2 Automatisation des connexions

A ce stade nous pouvons définir autant d'utilisateurs que souhaité. Chacun devra se connecter à MySQL avec la syntaxe `$ mysql -u utilisateur -p` (utilisateur étant à remplacer par le login). En réponse MySQL demandera le mot de passe correct.

Il peut paraître lourd, alors que LINUX vous a déjà demandé de vous identifier correctement au login, de se re-identifier à nouveau à chaque lancement d'un client MySQL.

Cette procédure est heureusement contournable.

A chaque lancement d'un client MySQL celui-ci va en effet vérifier la présence des fichiers suivants:

- `/etc/my.conf`
- `/var/lib/mysql/my.conf`
- `/.my.conf`, étant le répertoire home de l'utilisateur LINUX dûment logué et identifié.

Dès qu'il trouvera un fichier le client MySQL lira les données de configuration qui y sont rangées et ira à la recherche du fichier suivant dans la liste et dans cet ordre.

Les données les plus récentes iront à chaque fois écraser les plus anciennes ce qui permet de personnaliser très finement le comportement du client MySQL en fonction des besoins de l'utilisateur. Et parmi ces données, mais vous l'aurez déjà deviné, figurent le mot de passe de connexion au serveur MySQL !

Dans une installation de base aucun de ces fichiers n'existe. Par contre le sous-répertoire `/usr/share/mysql/` contient un certain nombre de fichiers exemples nommés `my-small.cnf`, `my-medium.cnf`, etc. qui correspondent à des propositions de fichiers de configuration pour différents cas d'utilisation. Les premières lignes de chacun de ses fichiers précisent les domaines d'application.

Pour automatiser le login MySQL pour un utilisateur lambda en se basant sur le modèle medium procéder comme suit :

- Copier `/usr/share/mysql/my-medium.cnf` vers `/.my.cnf` (fichier caché).
- Editer `/.my.cnf` et modifiez la rubrique `[client]`, la première qui soit proposée après l'entête, de telle sorte à faire apparaître une ligne `password = mot_de_passe` (remplacer `mot_de_passe` par le mot de

passer MySQL en clair de l'utilisateur). Dans le fichier exemple cette ligne est affectée d'un signe commentaire (#) qu'il faudra effacer. Pour simplifier vous pouvez même supprimer toutes les autres lignes de ce fichier et ne laisser que le paragraphe [client]. De la sorte vous évitez d'écraser des configurations ajustées qui auraient été mises en place, ou pourraient l'être ultérieurement, dans /etc/my.cnf ou /var/lib/mysql/my.cnf.

- Sauvegardez et modifiez les droits d'accès du fichier sauvegardé en tapant `$ chmod 700 .my.cnf` pour limiter les droits d'accès. Il serait en effet détestable qu'on puisse lire votre mot de passe MySQL !

Le tour est joué ! A la prochaine connexion avec le client MySQL celui-ci trouvera le mot de passe et établira une connexion directe en utilisant le login LINUX de l'utilisateur et le mot de passe mémorisé dans / .my .cnf. Pour vérifier tapez simplement `$ mysql` pour lancer un client MySQL. La connexion devrait être immédiate. Vérifiez sous quel login vous êtes entré en tapant `mysql> select user();` ce qui devrait provoquer l'affichage d'un tableau à une seule cellule titrée user() et contenant une information du genre lambda@localhost.

5 Contrôle des accès MySQL initiés par des scripts PHP.

Un script PHP qui accède à une base de données MySQL le fait dans des conditions fixées par la syntaxe de la commande de connexion. En PHP on obtient une connexion à un serveur MySQL au moyen de la commande `mysql_connect ()`. Celle-ci utilise trois arguments, \$host, \$user, \$password et renvoie une valeur TRUE, 1 si la connexion est établie, FALSE, 0 si celle-ci a échoué.

Le login de connexion ainsi que le mot de passe apparaissent donc en clair dans le script de connexion ce qui est moyennement satisfaisant. Aspect positif le script lui-même n'est jamais envoyé au client puisque PHP s'exécute entièrement sur le serveur (contrairement à JAVA). Il n'y a donc aucune raison pour qu'un visiteur puisse lire ce mot de passe. Oui, mais... il n'y a pas que des visiteurs ``normaux" !

Une protection supplémentaire (et aussi une simplification si on rédige beaucoup d'applications PHP) est de définir ses variables \$host, \$user et \$password dans un fichier extérieur au script lui-même et d'incorporer ces valeurs au moyen d'une directive include. Avantage : le fichier qui contient les variables peut être stocké en dehors des sous-répertoires consultés normalement par Apache (hors de l'arborescence /var/www/html/), et protégé par un fichier .htaccess.

Finalement le contrôle d'accès à MySQL via un script PHP découle directement de ce qui a été exposé au chapitre précédent et s'appuie directement sur les règles d'accès à MySQL définies dans la base de données /var/lib/mysql/mysql/. Il peut donc s'avérer souhaitable de créer un compte utilisateur MySQL spécifique aux applications PHP qui tournent sur le serveur. On pourrait aussi imaginer d'en créer plusieurs, 1 par famille de scripts par exemple.

6 Cas particulier de phpMyAdmin.

PhpMyAdmin est une petite merveille, un ensemble de scripts PHP, qui permet d'accéder au serveur MySQL en utilisant une interface graphique plus avenante que le spartiate client MySQL en mode texte.

L'installation par défaut conduit à faire de phpMyAdmin un client MySQL avec un login root sans mot de passe (d'où l'intérêt de ne pas mettre, dans un premier temps, de mot de passe au compte root faute de quoi phpMyAdmin ne serait plus en mesure d'accéder au serveur MySQL).

En fait le fonctionnement de phpMyAdmin est réglé par le fichier
/var/www/html/phpMyAdmin/config.inc.php3

Ce fichier, assez court, contient notamment les lignes suivantes :

```
// The $cfgServers array starts with $cfgServers[1]. Do
not use $cfgServers[0].

// You can disable a server config entry by setting host
to ''.

$cfgServers[1]['host'] = 'localhost'; // MySQL hostname

$cfgServers[1]['port'] = ''; // MySQL port - leave blank
for default port

$cfgServers[1]['adv_auth'] = false; // Use advanced
authentication?

$cfgServers[1]['stduser'] = 'root'; // MySQL standard user
(only needed with advanced auth)

$cfgServers[1]['stdpass'] = ''; // MySQL standard password
(only needed with advanced auth)

$cfgServers[1]['user'] = 'root'; // MySQL user (only
needed with basic auth)

$cfgServers[1]['password'] = ''; // MySQL password (only
needed with basic auth)

$cfgServers[1]['only_db'] = ''; // If set to a db-name,
only this db is accessible

$cfgServers[1]['verbose'] = ''; // Verbose name for this
host - leave blank to show the hostname
```

Le texte de ce fichier semble assez explicite. Il nécessite toutefois une analyse plus détaillée pour bien en comprendre certaines finesses.

\$cfgServers[1] désigne le premier serveur MySQL, le seul qui nous intéresse ici (il semblerait qu'il y ait des vicieux qui démarrent plusieurs serveurs.. imaginez un peu ;-)).

Le paramètre important est [adv_auth]. Il peut prendre deux valeurs, false (défaut à l'installation) et true.

6.1 Configuration avec adv_auth = false

Avec false l'authentification se fait selon une ancienne méthode qui utilise les valeurs de \$cfgServers[1]['user'] et \$cfgServers[1]['password'] pour s'identifier auprès de MySQL. L'installation de base résumée ci-dessus

reflète cette authentification. En clair phpMyAdmin s'enregistre sous compte root et sans mot de passe. C'est précisément la raison pour laquelle il ne faut surtout pas, en cours d'installation de MySQL, donner suite à la proposition du script de définir un mot de passe administrateur (voir http://jeanmarc.lichtle.free.fr/Apache_PHP_MySQL.html). Définir un mot de passe à ce stade de l'installation bloquerait le fonctionnement ultérieur de phpMyAdmin. Pour le vérifier faites simplement l'essai de fixer un mot de passe pour root dans MySQL. La syntaxe SQL est la suivante (mais vous l'aurez déjà deviné) :

```
mysql> update user set Password=password('mdp_root') where user = 'root';
```

Un petit coup de # `mysqladmin flush-privileges` (y'en a qui avaient oublié !?) et voilà le mot de passe administrateur pris en compte par le serveur MySQL. Pour la petite histoire c'est la dernière fois que vous tapez cette commande avec cette syntaxe simple. Maintenant que root a un mot de passe dans MySQL il faudra utiliser la syntaxe # `mysqladmin -u root -p flush-privileges`, ce qui va provoquer la demande du mot de passe par le serveur.

Et maintenant adieu phpMyAdmin, ça ne marche plus ! Seulement voilà, arrivé à ce stade de notre étude nous savons comment faire prendre en compte ce mot de passe par phpMyAdmin. Il suffit de l'incorporer au fichier de configuration `/var/www/html/phpMyAdmin/config.inc.php3` à la ligne `$cfgServers[1]['password'] = "`.

Au prochain lancement de phpMyAdmin le fichier de configuration va être pris en compte et tout va rentrer dans l'ordre, la connexion s'effectuant à nouveau de façon directe...

Bon, ça ne marche pas ? Ne vous affolez pas ! Je fais le pari que vous avez fait des essais de `.htaccess` ou de `httpd.conf` sur le sous répertoire de phpMyAdmin et qu'il reste des traces des essais effectués précédemment. Supprimez (ou renommez) l'éventuel fichier `.htaccess` situé dans le sous répertoire `/var/www/html/phpMyAdmin/` et/ou mettez des commentaires (#) aux éventuelles lignes ajoutées à `/etc/httpd/conf/httpd.conf`. Attention de relancer Apache si vous modifiez `httpd.conf` !

6.2 Configuration avec `adv_auth=true`

L'authentification avancée diffère légèrement du cas précédent en ce que le login et le mot de passe de l'utilisateur qui lance phpMyAdmin sont demandés au lancement. Il n'y a plus un login de connexion unique, celui-ci peut changer en fonction des informations données par l'utilisateur qui lance phpMyAdmin. Ces informations sont ensuite comparées à la base de données des utilisateurs autorisés par MySQL. La connexion est établie si les informations sont correctes.

Il faut toutefois établir une première connexion de façon à avoir un accès temporaire à la table `user`. C'est à cette fin que la configuration prévoit un compte défini par `$cfgServers[1]['stduser']` et `$cfgServers[1]['stdpass']`, ces deux variables devant permettre la vérification des droits d'accès. Il va sans dire que le `[stduser]` désigné ici devra avoir des droits suffisants pour lire la table `user`.

Cette authentification offre au minimum deux avantages :

- Les accès à MySQL via phpMyAdmin sont différenciés suivant les utilisateurs et suivent exactement les règles d'autorisation d'accès à MySQL.
- Le mot de passe administrateur n'a plus à figurer dans le fichier de configuration de phpMyAdmin. Il suffit de créer dans la table `user` de MySQL un utilisateur, nommé par exemple `phpMyAdmin`, dont les droits sont strictement limités à la consultation des tables (et donc de la table `user`). Il suffit de mettre un seul 'Y' et 13 'N' dans la fameuse requête de création d'un utilisateur. N'oubliez pas de

relancer `mysqladmin -u root -p flush-privileges` hein !

Pour le fun, et c'est une excursion vers l'étude de la configuration avancée des droits d'accès MySQL on peut même limiter encore plus les droits de cet utilisateur spécial. Il suffit de ne lui donner aucun droit dans la table `user` (14 fois 'N') et de lui créer une ligne dans la table `db`. Cherchez bien, la syntaxe est extrêmement proche de celle qui ajoute un utilisateur dans `user`, il suffit de préciser cette fois que ces droits s'applique spécifiquement à la base de données `mysql`. L'utilisateur 'phpMyAdmin' aura donc uniquement le droit de lire le contenu des tables `user`, `db` etc.. qui constituent la base `mysql`.

Vous pouvez même aller plus loin dans le raisonnement et limiter les droits strictement à la table `user`. Mais là c'est une autre affaire, je sens que vous allez passer quelques heures sur l'étude détaillée des droits d'accès MySQL.

7 Le petit bréviaire

Je ne sais pas si vous êtes comme moi mais j'ai besoin de prendre des notes pour retrouver rapidement les informations vitales dans un document aussi touffu que celui-ci.

Parmi les informations qu'il me semble intéressant de résumer ici je dresse à toutes fins utiles la liste des fichiers et répertoires qui ont été évoqués ci-dessus (dans leur ordre d'apparition en scène):

Répertoire /fichier	Fonction
<code>/var/www/html/</code>	Base arborescence site Web
<code>/var/www/html/phpMyAdmin/config.inc.php3</code>	Configuration de phpMyAdmin
<code>/var/lib/mysql/mysql/</code>	Base de données des droits d'accès
<code>/var/lib/mysql/my.conf</code>	Configuration de base de MySQL (optionnel)
<code>/etc/httpd/auth/</code>	Rangement des fichiers d'authentification par <code>.htaccess</code>
<code>/etc/httpd/conf/httpd.conf</code>	Configuration serveur Apache
<code>/etc/rc.d/init.d/httpd</code>	Lancement du serveur Apache option restart
<code>/etc/rc.d/init.d/mysql</code>	Lancement du serveur MySQL opt stop ou start
<code>/etc/my.conf</code>	Configuration de base de MySQL (optionnel)
<code>/.my.conf</code>	Configuration de base de MySQL (opt. spécif. util.)
<code>/usr/share/mysql/</code>	Contient des exemples de fichier <code>my.cnf</code>
<code>/var/log/httpd/</code>	Contient les fichiers log

Je précise à nouveau que ce document s'applique au cas d'une installation en environnement LINUX Mandrake 8.0 telle que décrite dans un document identifié plus haut. Un système qui serait basé sur une autre distribution et/ou qui utiliserait des versions d'Apache, PHP ou MySQL qui auraient été compilées par l'utilisateur pourrait utiliser d'autres répertoires pour stocker les différentes informations. L'essentiel est d'obtenir un ensemble cohérent.

8 Conclusion

L'exposé ci-dessus devait vous mettre en situation d'appréhender sérieusement la question assez complexe (mais oh combien stratégique) du contrôle des accès sur un serveur Apache PHP MySQL. J'espère avoir atteint cet objectif en montrant comment structurer la réflexion :

- protection du serveur Apache
- protection du serveur MySQL
- mise en place de contrôles d'accès via le langage PHP et plus précisément avec le script phpMyAdmin.

Il est évident qu'un sujet aussi vaste ne peut être qu'effleuré en quelques pages aussi je vous renvoie vers la littérature plus complète (livres, HowTo, pages man etc..) sur les différents points abordés.

J'espère sincèrement que ce document servira un jour à quelqu'un. Sa rédaction est de toute façon justifiée par le simple fait que l'exercice m'a obligé à mettre mes connaissances et mes notes au propre, à tester pas à pas toutes mes propositions. J'en suis donc le premier lecteur. N'hésitez pas à me faire part de votre avis sur l'intérêt de cette étude. Toute suggestion d'amélioration sera bienvenue.

L'auteur

JML dit Jean-Marc LICHTLE, email jean-marc.lichtle@gadz.org, ingénieur Arts et Métiers promo CH173 (rigolez pas, à l'époque les écrans n'existaient pas, le conversationnel c'était à grands renforts de télétypes et de rouleaux de papier! Pire encore, 16 clés, 16 voyants et un bouton run! Beurk.....)

Cette page est issue de la documentation 'pré-wiki' de Léa a été convertie avec HTML::WikiConverter. Elle fut créée par Jean-Marc LICHTLE le 01/10/2001.

Copyright

Copyright © 01/10/2001, Jean-Marc LICHTLE



*Ce document est publié sous licence Creative Commons
Attribution, Partage à l'identique, Contexte non commercial 2.0 :*
<http://creativecommons.org/licenses/by-nc-sa/2.0/fr/>