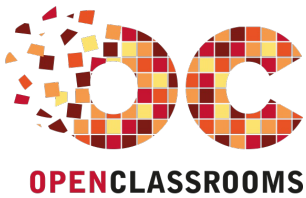
# Le Safe Mode sous PHP

**Par Zazou**




www.openclassrooms.com


*Dernière mise à jour le 19/12/2009*

# Sommaire

# Le Safe Mode sous PHP

Par Zazou
Mise à jour : 19/12/2009
Difficulté : Facile


Vous n'arrivez pas à uploader un fichier sur votre FTP *via* un script PHP ?
Vous n'arrivez pas à supprimer un fichier de votre FTP *via* un autre script PHP ?
En gros, vous n'arrivez pas à interagir avec des fichiers de votre FTP *via* PHP ?
Pourtant vous êtes certain de votre script, vos chemins, des noms de fichiers, etc. ?

Le Safe Mode est activé sur votre serveur ! Et on va résoudre tous vos problèmes ! En avant !

Sachez cependant que le Safe Mode est obsolète depuis PHP 5.3 et sera supprimé dès PHP 6.0.
Par conséquent, plus de souci à se faire à ce niveau-là si votre hébergeur utilise PHP 5.3 et +.
L'activation du Safe Mode sur un serveur sous PHP v5.3 et + génère une erreur `E_DEPRECATED` .

Sommaire du tutoriel :



## Qu'est-ce que le Safe Mode ?

Je vais faire simple, je vais vous donner la définition selon la doc PHP :

**Citation : La doc PHP**

> Le « Safe Mode » est le mode de sécurité de PHP : une solution au problème de partage de PHP sur un serveur. Ce système pêche au niveau de l'architecture car il n'est pas correct de tenter de résoudre ce problème au niveau de PHP, mais les solutions alternatives basées sur le serveur web et l'OS ne sont pas réalistes. De nombreux intervenants, notamment les fournisseurs d'hébergement, utilisent le « Safe Mode ».

En d'autres termes (parce que la doc n'explique pas suffisamment), quand le **Safe Mode** est activé, le serveur vérifie que vous êtes bien le propriétaire du fichier/dossier avant d'interagir avec lui, que ce soit pour de la simple lecture ou de la manipulation.

Ok, mais si ce fichier est sur mon FTP, j'en suis donc le propriétaire, non ?

D'une certaine manière oui, mais d'une autre non. Donc oui le FTP vous appartient. Mais sur tout système UNIX qui se respecte, il y a des droits sur les fichiers. Chaque utilisateur appartient à un groupe. C'est le principe même du CHMOD si vous le saviez déjà.

Prenons en exemple un réseau UNIX sur lequel tous les professeurs et étudiants d'une université peuvent se connecter. Chaque personne aura son propre UID. Les professeurs appartiendront au groupe *professeurs*, tandis que les étudiants au groupe *étudiants*. Ces deux groupes ont d'ailleurs un GID également ! Par conséquent, deux professeurs ou deux étudiants auront le même GID mais pas le même UID.

Le Safe Mode fonctionne sur cette base, l'UID cherchant à modifier le fichier doit être égale à celui à qui appartient le fichier. En utilisant les chemins relatifs basiques, l'UID du fichier est celui de l'utilisateur *root*, mais votre UID à vous (celui du script PHP en vérité) n'est pas le même que celui de *root*, l'accès est donc refusé.

Et PHP, plutôt que de vous retourner une erreur claire et simple à comprendre du genre « *Vous n'avez pas les droits suffisants pour interagir avec ce fichier/dossier* », ne vous dit rien et vous laisse chercher pendant des plombes d'où provient l'erreur !

Et grâce à ce tutoriel, votre calvaire se termine enfin. Avant de voir comment résoudre ces problèmes de Safe Mode, nous allons voir les conséquences de son activation.

## Les conséquences du Safe Mode

Il y a trois conséquences possibles :

- soit la fonction est désactivée purement et simplement OU n'a plus d'effet ;
- soit la fonction est limitée ;
- soit la fonction n'opère aucun changement.

Comme la 3e conséquence ne nous intéresse pas ici, je ne vous donnerai que les deux premières.

### *Les fonctions purement et simplement désactivées ou qui n'ont plus d'effet*

| Fonction | Conséquence |
|---|---|
| dl() | Désactivée |
| shell_exec() | Désactivée |
| set_time_limit() | N'as plus d'effet |
| max_execution_time() | N'as plus d'effet |
| opérateur guillemets obliques ( ` : touche 7 du clavier ) | Désactivé |

### *Les fonctions limitées*

| Fonction | Conséquence |
|---|---|
| dbmopen() | UID du fichier doit être égale à UID du script |
| dbase_open() | UID du fichier doit être égale à UID du script |
| filepro() | UID du fichier doit être égale à UID du script |
| filepro_rowcount() | UID du fichier doit être égale à UID du script |
| filepro_retrieve() | UID du fichier doit être égale à UID du script |
| pg_lo_import() | UID du fichier doit être égale à UID du script |
| posix_mkfifo() | UID du fichier doit être égale à UID du script |
| putenv() | UID du fichier doit être égale à UID du script |
| move_uploaded_file() | UID du fichier doit être égale à UID du script |
| chdir() | UID du fichier doit être égale à UID du script |
| exec() | Ne peut exécuter que les programmes présents dans le safe_mode_exec_dir |
| system() | Ne peut exécuter que les programmes présents dans le safe_mode_exec_dir |
| passthru() | Ne peut exécuter que les programmes présents dans le safe_mode_exec_dir |
| popen() | Ne peut exécuter que les programmes présents dans le safe_mode_exec_dir |
| fopen() | UID du fichier doit être égale à UID du script |
| mkdir() | UID du fichier doit être égale à UID du script |
| rmdir() | UID du fichier doit être égale à UID du script |
| rename() | UID du fichier doit être égale à UID du script |

| | |
|---|---|
| unlink() | UID du fichier doit être égale à UID du script |
| copy() | UID du fichier doit être égale à UID du script |
| chgrp() | UID du fichier doit être égale à UID du script |
| chown() | UID du fichier doit être égale à UID du script |
| chmod() | UID du fichier doit être égale à UID du script |
| touch() | UID du fichier doit être égale à UID du script |
| symlink() | UID du fichier doit être égale à UID du script |
| link() | UID du fichier doit être égale à UID du script |
| highlight_file() | UID du fichier doit être égale à UID du script |
| show_source() | UID du fichier doit être égale à UID du script |
| parse_ini_file() | UID du fichier doit être égale à UID du script |
| apache_request_headers() | Les en-têtes commençant par *authorization* (sensibles à la casse) ne seront pas retournés |
| link() | UID du fichier doit être égale à UID du script |
| header() | L'UID du script est ajouté à la partie *realm* de l'en-tête WWW-Authenticate |
| variables PHP_AUTH | Indisponible dans la *superglobal* $_SERVER |
| mail() | Le 5e paramètre est désactivé (drapeaux/paramètres optionnels) |
| session_start() | Le propriétaire d'un script doit être le même que celui du répertoire session.save_path, si le répertoire par défaut session.save_handler est utilisé |
| Toutes les fonctions sur les flux et sur le système de fichiers | UID du fichier doit être égale à UID du script |

## Contournons le Safe Mode

Voilà, après du blabla intempestif, entrons dans le vif du sujet ! La résolution du problème !

### *La théorie*

Pour résoudre le problème, c'est très simple. Il suffit de manipuler des fichiers dont le script est propriétaire ! L'UID du script doit correspondre à l'UID du fichier. Pour ce faire, il faut utiliser **le chemin canonique absolu** du fichier.

Quel nom barbare ! 

Autrement dit, il faut utiliser le **VRAI** chemin du fichier. Le chemin qui dit que le fichier, c'est le nôtre !

### *La pratique*

Pour connaître le chemin canonique absolu d'un fichier, il suffit de placer ce fichier (que l'on appellera chemin.php) dans le dossier dans lequel on souhaite manipuler les fichiers. Il ne reste qu'à l'appeler dans son navigateur favori.

**Code : PHP - chemin.php**

```php
<?php
echo realpath('chemin.php');
```

Ce code retournera quelquechose comme : /home/login/[...]/chemin.php où [...] peut valoir 1 ou plusieurs descentes de dossier. Le voilà, notre chemin canonique absolu : /home/login/[...]/ ; c'est avec lui que l'on devra toujours travailler pour manipuler des fichiers si le Safe Mode est actif.

### *Exemple concret : un script d'upload d'avatars*

Sur mon FTP à la racine, je crée un dossier `images`, puis dans celui-ci un dossier `avatars`. J'ai donc une arborescence telle que celle-ci :

- racine
  - images
    - avatars

Mon script d'upload d'avatars devra utiliser la fonction `move_uploaded_file()` , qui ressemblera à ceci :

*(J'ai volontairement réduit le code à sa plus simple utilisation, l'idée n'étant pas de vous apprendre à uploader des fichiers mais bien de contourner le Safe Mode ).*

**Code : PHP**

```php
<?php
$tmp_name = $_FILES['avatar']['tmp_name']; // Le fichier temporaire
$nomFichier = basename($_FILES['avatar']['name']); // Le nom du
fichier
$chemin_canonique = '/home/Zazou/images/avatars/';
// Mon FTP s'appelle Zazou qui correspond simplement à mon login de
connexion à mon FTP
// J'ai créé à la racine un dossier images, et dans celui-ci un
dossier avatars (cf. arborescence)
$destination = $chemin_canonique.$nomFichier; // On formate la
destination
move_uploaded_file($tmp_name,$destination); // On uploade
```

Il ne faut pas oublier que le CHMOD du dossier `avatars` doit être à 777.

Et voilà : le tour est joué ! Si toutes les conditions sont bien remplies, vous ne devriez plus avoir de problèmes.
Eh voilà, ce tutoriel est fini. J'espère qu'il vous aura aidé à résoudre tous vos soucis d'upload ou autres. Vous avez désormais toutes les cartes en main pour créer vos scripts d'upload ou pour utiliser toutes les fonctions qui sont limitées par le Safe Mode. Je ne veux donc plus entendre des « Bouhouhouh, ça marche pas ! J'vais me suicider ». Je vous souhaite bonne chance, en espérant ne pas vous revoir sur les forums avec un problème lié au Safe Mode.

Merci à Fihld et Barbatos pour la zCorrection