Dépôt de fichiers

Alexandre Niveau

GREYC — Université de Caen

En partie adapté du cours de Jean-Marc Lecarpentier

Dépôt de fichiers ou *upload*

- Envoi d'un fichier local vers le serveur
- Faisable en HTTP avec un formulaire POST
- Standardisé en 1995 (RFC 1867 [http://tools.ietf.org/html/rfc1867])
- Utile pour de nombreuses applications, par exemple pour un webmail (attacher une pièce jointe) et pour l'immense majorité des sites web dits « 2.0 » (images, vidéos, son…)

Le formulaire HTML

- Élément `input` de type `file`, donne un widget « Parcourir » :

  Parcourir… Aucun fichier sélectionné.

- Ne marche qu'avec la méthode POST
- Il faut préciser dans l'élément `form` que le type MIME des données est `multipart/form-data` (par défaut, c'est `application/x-www-form-urlencoded`), sinon le serveur ne comprendra pas le contenu du formulaire
- Exemple :

```
<form enctype="multipart/form-data" action="traitement.php"
                                    method="POST">

    <input type="file" name="pj">
    <input type="file" name="avatar">

    <button type="submit">Valider</button>

</form>
```

Traitement du fichier en PHP

- PHP stocke les informations des fichiers uploadés dans le tableau `$_FILES`
- Chaque `input` de type `file` correspond à une case du tableau (indexée par le `name` de l'`input`) : `$_FILES['pj']`, `$_FILES['avatar']`…
- Chaque case du tableau est elle-même un tableau avec les informations du fichier (nom d'origine, emplacement sur le serveur, taille, etc.)

- Le cours ne rentrera pas dans les détails, voir le manuel [http://www.php.net/manual/fr/features.file-upload.php] pour en savoir plus

Les clés du tableau

- `$_FILES['pj']['name']` : nom du fichier sur la machine du client
- `$_FILES['pj']['tmp_name']` : chemin complet du fichier uploadé sur le serveur (dans le répertoire `/tmp` par défaut)
- `$_FILES['pj']['size']` : taille en octets du fichier envoyé
- `$_FILES['pj']['type']` : type MIME du fichier, si transmis (par exemple `image/jpeg`) ; non vérifié par PHP !
- `$_FILES['pj']['error']` : code d'erreur associé à la transmission (PHP>4.2.0)

Les codes d'erreur

- `UPLOAD_ERR_OK` (valeur 0) : succès de l'upload
- `UPLOAD_ERR_INI_SIZE` (valeur 1) taille du fichier supérieure à la valeur de l'option PHP `upload_max_filesize`
- `UPLOAD_ERR_FORM_SIZE` (valeur 2) taille du fichier supérieure à la valeur `MAX_FILE_SIZE` spécifiée dans le formulaire HTML (obsolète et non fiable)
- `UPLOAD_ERR_PARTIAL` (valeur 3) fichier partiellement uploadé
- `UPLOAD_ERR_NO_FILE` (valeur 4) aucun fichier uploadé
- `UPLOAD_ERR_NO_TMP_DIR` (valeur 6) pas de répertoire temporaire (PHP>5.0.3)
- `UPLOAD_ERR_CANT_WRITE` (valeur 7) impossible d'écrire sur le disque du serveur (PHP>5.1.0)
- `UPLOAD_ERR_EXTENSION` (valeur 8) une extension PHP a arrêté l'envoi de fichier, mais on ne peut pas savoir laquelle... (PHP>5.2.0)

Traitement du fichier

- Le fichier uploadé est placé sur le serveur dans un répertoire temporaire (par défaut `/tmp`) sous un nom quelconque (par ex. `/tmp/php7Mhh3j`)
- Il faut déplacer le fichier vers le répertoire où on veut le conserver, et le renommer
- On peut utiliser la fonction `move_uploaded_file` (PHP>4.0.3), qui vérifie que le fichier à déplacer résulte bien d'un upload (plus sécurisé qu'un simple `copy`)
- La fonction prend deux paramètres : le nom complet du fichier temporaire, et le nom complet du fichier de destination
- Renvoie `true` en cas de succès et `false` sinon (soit le fichier n'est pas un fichier uploadé valide, soit le déplacement a échoué)

Exemple

```
<?php
```

```php
if (move_uploaded_file($_FILES['pj']['tmp_name'], "/users/16715771/toto/
uploadedFile.txt")) {
  echo "Copie de fichier réussie";
} else {
  echo "Problème de copie de fichier";
}
?>
```

Le problème du dépôt de fichiers

- Comme toujours, se méfier de ce qui vient du client !
- Si vous rendez accessible sur le web des fichiers déposés par les internautes, il faut être très prudent : par ex., si le fichier est un script PHP, l'internaute peut exécuter du code sur votre serveur…

  ➤ les fichiers déposés doivent être dans un répertoire à part, configuré pour que les fichiers qu'il contient ne soient jamais exécutés

Application de la règle d'or

- Ne pas s'appuyer sur les informations données par le client :
  - l'extension du fichier et le type MIME indiqué peuvent être faux : il faut vérifier que le type du fichier est bien celui attendu. Pour une image, on utilisera par ex. la fonction PHP `exif_imagetype` [http://php.net/manual/fr/function.exif-imagetype.php], qui analyse le *contenu* du fichier pour détecter le format de l'image (et renvoie `false` si ce n'est pas une image).
  - ne pas utiliser tel quel le nom original du fichier (`$_FILES['fichier']['name']`) : utiliser son propre schéma de nommage (à base d'identifiants aléatoires), et mettre l'extension qui correspond à ce qu'on a détecté. Si besoin, on peut toujours garder le nom original dans la base de données.

Prudence !

- **Attention** : il est possible de créer une image parfaitement valide qui exécute du code quand on la donne à l'interpréteur PHP !

  ➤ si on a suivi tous les conseils précédents, ça ne pose pas de problème, mais si on veut être vraiment sûr, une possibilité est de recréer l'image (avec ImageMagick ou GD) pour éliminer les métadonnées (mais la qualité de l'image peut baisser)
- Si vraiment vous tenez à ce que le fichier ait un nom proche de l'original (déconseillé : il est très délicat de penser à toutes les failles possibles…), le strict minimum est :
  - mettre la bonne extension
  - tronquer le nom à une longueur raisonnable

[hidden-php.jpg]
Une image JPEG dont les métadonnées EXIF contiennent du PHP. Essayez de l'exécuter !

    - décider d'un ensemble de caractères autorisés, et éliminer tous les caractères non autorisés. Par exemple, on autorise `[A-Za-z0-9_]` (caractères alphanumériques ASCII + underscore) et on remplace tout le reste par _.

- Exemple : la *double extension attack* consiste à uploader un script PHP avec un nom comme `toto.php.jpg`. Le serveur web Apache était souvent configuré pour qu'un tel fichier soit exécuté comme du PHP.

  ➤ les vecteurs d'attaques peuvent être très surprenants, on ne peut pas tout prévoir. Il faut laisser à l'internaute le moins de contrôle possible sur le fichier uploadé.

Configuration de PHP

- Pour que l'upload fonctionne correctement, plusieurs options à vérifier dans la configuration de PHP :
    - `file_uploads` doit être à 1
    - `max_execution_time` (temps maximal d'exécution du script) et `max_input_time` (temps maximal de réception des données) : attention aux gros fichiers, aux fichiers multiples, et aux connexions lentes
    - `memory_limit` mémoire maximum allouée à un script
    - `post_max_size` taille maximum des données transmises par un formulaire
    - `upload_max_filesize` taille maximum pour un upload de fichier
    - `upload_tmp_dir` répertoire où sont placés les fichiers envoyés (`/tmp` par défaut)
- **Attention** : si le total des données POST est > `post_max_size`, rien n'est transmis au serveur. Les tableaux `$_POST` et `$_FILES` sont **vides**, donc `$_FILES['pj']['error']` n'existe même pas !

Spécifications et normes

- RFC 1867 — Form-based File Upload in HTML [http://tools.ietf.org/html/rfc1867]

Références et guides

- Manuel PHP sur l'upload [http://www.php.net/manual/fr/features.file-upload.php]