PHP文件上传

POST 方法上传

本特性可以使用户上传文本和二进制文件。用 PHP 的认证和文件操作函数，可以完全控制允许哪些人上传以及文件上传后怎样处理。

PHP 能够接受任何来自符合 RFC-1867 标准的浏览器上传的文件。

> **注意: 相关的设置**
>
> 请参阅 php.ini 的 file_uploads，upload_max_filesize，upload_tmp_dirpost_max_size 以及 max_input_time 设置选项。

请注意 PHP 也支持 PUT 方法的文件上传，Netscape Composer 和 W3C 的 Amaya 客户端使用这种方法。请参阅对 PUT 方法的支持以获取更多信息。

示例 #1 文件上传表单

可以如下建立一个特殊的表单来支持文件上传：

```
<!-- The data encoding type, enctype, MUST be specified as below -->
<form enctype="multipart/form-data" action="__URL__" method="POST">
    <!-- MAX_FILE_SIZE must precede the file input field -->
    <input type="hidden" name="MAX_FILE_SIZE" value="30000" />
    <!-- Name of input element determines name in $_FILES array -->
    Send this file: <input name="userfile" type="file" />
    <input type="submit" value="Send File" />
</form>
```

以上范例中的 `__URL__` 应该被换掉，指向一个真实的 PHP 文件。

`MAX_FILE_SIZE` 隐藏字段（单位为字节）必须放在文件输入字段之前，其值为接收文件的最大尺寸。这是对浏览器的一个建议，PHP 也会检查此项。在浏览器端可以简单绕过此设置，因此不要指望用此特性来阻挡大文件。实际上，PHP 设置中的上传文件最大值是不会失效的。但是最好还是在表单中加上此项目，因为它可以避免用户在花时间等待上传大文件之后才发现文件过大上传失败的麻烦。

> **注意:**
>
> 要确保文件上传表单的属性是 `enctype="multipart/form-data"`，否则文件上传不了。

全局变量 $_FILES 包含有所有上传的文件信息。 数组的内容来自以下范例表单。我们假设文件上传字段的名称如下例所示，为 *userfile*。名称可随意命名。

- $_FILES['userfile']['name']

  客户端机器文件的原名称。

- $_FILES['userfile']['type']

文件的 MIME 类型，如果浏览器提供此信息的话。一个例子是“`image/gif`”。不过此 MIME 类型在 PHP 端并不检查，因此不要想当然认为有这个值。

- $_FILES['userfile']['size']

  已上传文件的大小，单位为字节。

- $_FILES['userfile']['tmp_name']

  文件被上传后在服务端储存的临时文件名。

- $_FILES['userfile']['error']

  和该文件上传相关的错误代码。

文件被上传后，默认地会被储存到服务端的默认临时目录中，除非 php.ini 中的 upload_tmp_dir 设置为其它的路径。服务端的默认临时目录可以通过更改 PHP 运行环境的环境变量 TMPDIR 来重新设置，但是在 PHP 脚本内部通过运行 putenv() 函数来设置是不起作用的。该环境变量也可以用来确认其它的操作也是在上传的文件上进行的。

示例 #2 验证上传的文件

以下范例处理由表单提供的文件上传。

```
<?php
$uploaddir = '/var/www/uploads/';
$uploadfile = $uploaddir . basename($_FILES['userfile']['name']);

echo '<pre>';
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
    echo "File is valid, and was successfully uploaded.\n";
} else {
    echo "Possible file upload attack!\n";
}

echo 'Here is some more debugging info:';
print_r($_FILES);

print "</pre>";

?>

1.上传了PHP文件
2.执行PHP文件


1.可以上传PHP文件
2.PHP文件可以执行
3.知道文件路径

防御方案：
1.黑名单；（不能有php文件）
2.限制类型
3.修改权限
```

```
防御方案总结：
1.文件后缀名限制；白名单限制（jpg,png,gif）
2.文件上传目录下的所有文件都没有可执行权限；
3.不返回上传文件路径；
4.修改上传文件的文件名；（随机文件名）
```

接受上传文件的 PHP 脚本为了决定接下来要对该文件进行哪些操作，应该实现任何逻辑上必要的检查。

例如可以用 $_FILES['userfile']['size'] 变量来排除过大或过小的文件，也可以通过 $_FILES['userfile']['type'] 变量来排除文件类型和某种标准不相符合的文件，但只把这个当作一系列检查中的第一步，因为此值完全由客户端控制而在 PHP 端并不检查。

同时，还可以通过 $_FILES['userfile']['error'] 变量来根据不同的错误代码来计划下一步如何处理。不管怎样，要么将该文件从临时目录中删除，要么将其移动到其它的地方。

如果表单中没有选择上传的文件，则 PHP 变量 $_FILES['userfile']['size'] 的值将为 0，$_FILES['userfile']['tmp_name'] 将为空。

如果该文件没有被移动到其它地方也没有被改名，则该文件将在表单请求结束时被删除。

示例 #3 上传一组文件

PHP 的 HTML 数组特性甚至支持文件类型。

```
<form action="" method="post" enctype="multipart/form-data">
<p>Pictures:
<input type="file" name="pictures[]" />
<input type="file" name="pictures[]" />
<input type="file" name="pictures[]" />
<input type="submit" value="Send" />
</p>
</form>
```

```
<?php
foreach ($_FILES["pictures"]["error"] as $key => $error) {
    if ($error == UPLOAD_ERR_OK) {
        $tmp_name = $_FILES["pictures"]["tmp_name"][$key];
        // basename() may prevent filesystem traversal attacks;
        // further validation/sanitation of the filename may be appropriate
        $name = basename($_FILES["pictures"]["name"][$key]);
        move_uploaded_file($tmp_name, "data/$name");
    }
}
?>
```

File upload progress bar can be implemented using Session Upload Progress.

错误信息说明

PHP 将随文件信息数组一起返回一个对应的错误代码。该代码可以在文件上传时生成的文件数组中的 `error` 字段中被找到，也就是 $_FILES['userfile']['error']。

- **`UPLOAD_ERR_OK`**

  其值为 0，没有错误发生，文件上传成功。
- **`UPLOAD_ERR_INI_SIZE`**

  其值为 1，上传的文件超过了 php.ini 中 upload_max_filesize 选项限制的值。
- **`UPLOAD_ERR_FORM_SIZE`**

  其值为 2，上传文件的大小超过了 HTML 表单中 *MAX_FILE_SIZE* 选项指定的值。
- **`UPLOAD_ERR_PARTIAL`**

  其值为 3，文件只有部分被上传。
- **`UPLOAD_ERR_NO_FILE`**

  其值为 4，没有文件被上传。
- **`UPLOAD_ERR_NO_TMP_DIR`**

  其值为 6，找不到临时文件夹。PHP 5.0.3 引进。
- **`UPLOAD_ERR_CANT_WRITE`**

  其值为 7，文件写入失败。PHP 5.1.0 引进。
- **`UPLOAD_ERR_EXTENSION`**

  Value: 8; A PHP extension stopped the file upload. PHP does not provide a way to ascertain which extension caused the file upload to stop; examining the list of loaded extensions with phpinfo() may help. Introduced in PHP 5.2.0.

常见缺陷

对 `MAX_FILE_SIZE` 设置的值，不能大于 php.ini 文件 设置中 upload_max_filesize 选项设置的值。其默认值为 2M 字节。

如果内存限制设置被激活，可能需要将 memory_limit 设置的更大些，请确认 memory_limit 的设置足够的大。

如果 max_execution_time 设置的值太小，脚本运行的时间可能会超过该设置。因此，也请保证 `max_execution_time` 足够的大。

> **注意**: max_execution_time 仅仅只影响脚本本身运行的时间。任何其它花费在脚本运行之外的时间，诸如用函数 system() 对系统的调用、sleep() 函数的使用、数据库查询、文件上传等，在计算脚本运行的最大时间时都不包括在内。

警告

max_input_time 以秒为单位设定了脚本接收输入的最大时间，包括文件上传。对于较大或多个文件，或者用户的网速较慢时，可能会超过默认的 `60 秒`。

如果 post_max_size 设置的值太小，则较大的文件会无法被上传。因此，请保证 `post_max_size` 的值足够的大。

As of PHP 5.2.12, the max_file_uploads configuration setting controls the maximum number of files that can uploaded in one request. If more files are uploaded than the limit, then $_FILES will stop processing files once the limit is reached. For example, if max_file_uploads is set to `10`, then $_FILES will never contain more than 10 items.

不对正在操作的文件进行验证可能意味着用户能够访问其它目录下的敏感信息。

请注意 CERN httpd 似乎会丢弃它从客户端获得的 content-type mime 头信息中第一个空格后所有的内容，基于这一点，CERN httpd 不支持文件上传特性。

鉴于文件路径的表示方法有很多种，我们无法确保用使用各种外语的文件名（尤其是包含空格的）能够被正确的处理。

开发人员不应将普通的 `input` 输入字段和文件上传的字段混用同一个表单变量（例如 `input` 名都用 `foo[]`）。

传多个文件

可以对 `input` 域使用不同的 `name` 来上传多个文件。

PHP 支持同时上传多个文件并将它们的信息自动以数组的形式组织。要完成这项功能，需要在 HTML 表单中对文件上传域使用和多选框与复选框相同的数组式提交语法。

示例 #1 上传多个文件

```
<form action="file-upload.php" method="post" enctype="multipart/form-data">
  Send these files:<br />
  <input name="userfile[]" type="file" /><br />
  <input name="userfile[]" type="file" /><br />
  <input type="submit" value="Send files" />
</form>
```

当以上表单被提交后，数组 *$FILES'userfile']*，*[*$FILES['userfile']['name'] 和 $_FILES['userfile']['size'] 将被初始化。

例如，假设名为 /home/test/review.html 和 /home/test/xwp.out 的文件被提交，则 $_FILES['userfile']['name'][0] 的值将是 review.html，而 $_FILES['userfile']['name'][1] 的值将是 xwp.out。类似的，$_FILES['userfile']['size'][0] 将包含文件 review.html 的大小，依此类推。

此外也同时设置了 $_FILES['userfile']['name'][0]， $_FILES['userfile']['tmp_name'][0]， $_FILES['userfile']['size'][0] 以及 $_FILES['userfile']['type'][0]。

警告

As of PHP 5.2.12, the max_file_uploads configuration setting acts as a limit on the number of files that can be uploaded in one request. You will need to ensure that your form does not try to upload more files in one request than this limit.

对 PUT 方法的支持

PHP 对部分客户端具备的 HTTP PUT 方法提供了支持。PUT 请求比文件上传要简单的多，它们一般的形式为：

```
PUT /path/filename.html HTTP/1.1
```

这通常意味着远程客户端会将其中的 /path/filename.html 存储到 web 目录树。让 Apache 或者 PHP 自动允许所有人覆盖 web 目录树下的任何文件显然是很不明智的。因此，要处理类似的请求，必须先告诉 web 服务器需要用特定的 PHP 脚本来处理该请求。在 Apache 下，可以用 *Script* 选项来设置。它可以被放置到 Apache 配置文件中几乎所有的位置。通常我们把它放置在 `<Directory>` 区域或者 `<VirtualHost>` 区域。可以用如下一行来完成该设置：

```
Script PUT /put.php
```

这将告诉 Apache 将所有对 URI 的 PUT 请求全部发送到 put.php 脚本，这些 URI 必须和 PUT 命令中的内容相匹配。当然，这是建立在 PHP 支持 .php 扩展名，并且 PHP 已经在运行的假设之上。 The destination resource for all PUT requests to this script has to be the script itself, not a filename the uploaded file should have.

With PHP you would then do something like the following in your put.php. This would copy the contents of the uploaded file to the file myputfile.ext on the server. You would probably want to perform some checks and/or authenticate the user before performing this file copy.

示例 #1 保存 HTTP PUT 文件

```php
<?php
/* PUT data comes in on the stdin stream */
$putdata = fopen("php://input", "r");

/* Open a file for writing */
$fp = fopen("myputfile.ext", "w");

/* Read the data 1 KB at a time
   and write to the file */
while ($data = fread($putdata, 1024))
  fwrite($fp, $data);

/* Close the streams */
fclose($fp);
```

```
fclose($putdata);
?>
```