

The Evils of 777

Joel Clermont
@jclermont

Refresher on permissions

Octal notation

```
  7      7      7
user  group world
r+w+x r+w+x r+w+x
4+2+1 4+2+1 4+2+1 = 777
```

Files versus directories

Purposely ignoring Windows on this issue

Bad advice abounds



concrete5[®]

[About](#)

[Community](#)

[Developers](#)

[Marketplace](#)

[Services](#)

[Documentation](#)

[Basic Setup](#)

[File/Directory Permissions](#)



[Upgrading concrete5](#)

[Internationalization](#)

[Installing on Specific Hosts](#)

[Moving a Site](#)

[Installation Help](#)

`/files/` and all items within (both files and directories) should be set to be readable and writable by the web server. That can be accomplished in the following ways, from best to worst:

1. If your server supports running as suexec/phpsuexec, the files should be owned by your user account, and set as 755 on all of them. That means that your web server process can do anything it likes to them, but nothing else can (although everyone can view them, which is expected.)
2. If 1 isn't possible, another good option is to set the apache user (either "apache" or "nobody") as having full rights to these files
3. If neither 1 or 2 are possible, `chmod 777` to `files/` and all items within (e.g. `chmod -R 777 file/*`)

This must include the `cache_objects` directory, or you're going to get all kinds of strange behavior with your site.

[« Previous](#)

[Next »](#)

Bad advice abounds



Correct Chmod Values

- config.php
 - 666 before installation
 - 644 after installation
- All other files - 644
- All directories - 755
 - There are some exceptions when it comes to directory permissions,
 - The *files* directory - 777
 - The *cache* directory - 777
 - The *store* directory - 777
 - The *images/avatars/upload* directory - 777

It is getting better though

The "big 3" CMSes no longer encourage 777

- see link on first slide from WordPress
- Joomla and Drupal also give big scary warnings

Why do some programs want 777?

It all depends on who the user is at the moment . . .

- FTP user?
- apache user?

. . . and how Apache is configured to run PHP scripts

- mod_php?
- CGI/FastCGI/FPM?

User uploads / automated installs or updates / caches

What is the danger of 777?

Think about what "world" writeable means . . . it's bad

- multi-tenant shared server is really bad
- dedicated server is less bad

A user of site 1 writes a malicious file to site 2

A user of site 1 reads a config file from site 2

- API keys, database credentials

Others?

Anyone not convinced yet?

Solution

Get PHP code to execute as your FTP user

FPM (FastCGI Process Manager) <http://php-fpm.org/>

- bundled with PHP as of 5.3.3 (July 2010)
- even nicer than running FastCGI alone

Another option

suPHP is a different approach to the same problem

- <http://www.suphp.org>

Less of a good idea, especially with FPM baked into PHP

Not necessarily an easy change

Requires server support

- can be tricky to get this if not natively offered

Zend Server woes

- no FastCGI support on Linux

Interesting reading on the topic:

http://weierophinney.net/matthew/archives/243-Running-mod_php-and-FastCGI-side-by-side.html

Temporary measures

Only use 777 outside of your public web root

Disable PHP in 777 folders
- or whitelist approved extensions

This should only hold you over until you get a real fix

Questions, comments or war stories?