13PermissionDenied

(13) Permission Denied

Error 13 indicates a filesystem permissions problem. That is, Apache was denied access to a file or directory due to incorrect permissions. It does not, in general, imply a problem in the Apache configuration files.

In order to serve files, Apache must have the proper permission granted by the operating system to access those files. In particular, the `User` or `Group` specified in `httpd.conf` must be able to read all files that will be served and*_ search the directory containing those files, along with all parent directories up to the root of the filesystem_*.

Typical permissions on a unix-like system for resources not owned by the `User` or `Group` specified in `httpd.conf` would be 644 `rw-r--r--` for ordinary files and 755 `drwxr-x-r-x` for directories or CGI scripts. You may also need to check extended permissions (such as SELinux permissions) on operating systems that support them.

If you are running 2.4, the AH error code may give you more information here.

- **AH00132**: file permissions deny server access
- **AH00035**: access denied because search permissions are missing on a component of the path

An Example

Lets say that you received the `Permission Denied` error when accessing the file `/usr/local/apache2/htdocs/foo/bar.html` on a unix-like system.

First check the existing permissions on the file:

```
cd /usr/local/apache2/htdocs/foo
ls -l bar.htm
```

Fix them if necessary:

```
chmod 644 bar.html
```

Then do the same for the directory and each parent directory (`/usr/local/apache2/htdocs/foo`, `/usr/local/apache2/htdocs`, `/usr/local/apache2`, `/usr/local`, `/usr`):

```
ls -la
chmod +x .
cd ..
# repeat up to the root
```

On some systems, the utility `namei` can be used to help find permissions problems by listing the permissions along each component of the path:

```
namei -m /usr/local/apache2/htdocs/foo/bar.html
```

If your system doesn't have namei, you can use parsepath. It can be obtained from here.

If all the standard permissions are correct and you still get a `Permission Denied` error, you should check for extended-permissions. For example you can use the command `setenforce 0` to turn off SELinux and check to see if the problem goes away. If so, `ls -alZ` can be used to view SELinux permission and `chcon` to fix them.

In rare cases, this can be caused by other issues, such as a file permissions problem elsewhere in your apache2.conf file. For example, a WSGIScriptAlias directive not mapping to an actual file. The error message may not be accurate about which file was unreadable.

DO NOT set files or directories to mode 777, even "just to test", even if "it's just a test server". The purpose of a test server is to get things right in a safe environment, not to get away with doing it wrong. All it will tell you is if the problem is with files that actually exist.

CGI scripts

Although the CGI script permission might look correct, the actual binary specified in the shebang might not have the proper permissions to be run. (Or some directory on its path, check with namei as explained above.)

(13)Permission denied: proxy: HTTP: attempt to connect to 127.0.0.1:8080 (localhost) failed

This error is not really about file permissions or anything like that. What it actually means is that httpd has been denied permission to connect to that IP address and port.

The most common cause of this is SELinux not permitting httpd to make network connections.

To resolve it, you need to change an SELinux boolean value (which will automatically persist across reboots). You may also want to restart httpd to reset the proxy worker, although this isn't strictly required.

```
# setsebool -P httpd_can_network_connect 1
```

For more information on how SELinux can affect httpd, read the httpd_selinux man page.