Security

- wp-content und wp-includes zu schützen?

wp-content und wp-includes zu schützen?

Kurze Antwort: Teilweise sinnvoll, aber mit wichtigen Einschränkungen

Es macht **eingeschränkt Sinn**, aber man muss unterscheiden, was genau "geschützt" wird – ein Pauschalblock kann die Seite auch kaputt machen.

wp-includes schützen meist unproblematisch

- `wp-includes` enthält WordPress-Core-Dateien (PHP-Klassen, JS, CSS), die **nicht direkt aufgerufen** werden müssen – sie werden von WordPress selbst intern geladen.
- Typische Regel (Apache/.htaccess):

```
RewriteRule ^wp-includes/[^/]+\.php$ - [F,L]
```

- **Nutzen:** Verhindert direktes Ausführen von PHP-Skripten in `wp-includes`, was ein gängiger Angriffsvektor ist (z.B. wenn eine Lücke eine Datei dort hochlädt/ausnutzt).
- **Risiko:** minimal – manche Plugins/Themes greifen selten direkt auf JS/CSS-Dateien in `wp-includes` zu (z.B. `wlwmanifest.xml`), daher sollte man PHP-Dateien blocken, nicht den ganzen Ordner inkl. Assets.

wp-content schützen ⚠ vorsichtiger, weil dort "lebende" Inhalte liegen

`wp-content` ist kein einheitlicher Ordner – er enthält:

Unterordner	Zweck	Direkter Zugriff nötig?
`uploads/`	Bilder, PDFs, Medien	**Ja**, muss öffentlich erreichbar sein
`themes/`	CSS/JS/Bilder des Themes	**Ja**, für Assets
`plugins/`	Plugin-Code, teils Assets	**Ja** für Assets, **Nein** für PHP direkt
`uploads/*.php`	Sollte es gar nicht geben	Wird oft als Malware-Ablage genutzt

Sinnvoll ist daher **nicht** "wp-content sperren", sondern gezielt:

```
# PHP-Ausführung in uploads verhindern (häufigster Angriffsweg)
<Directory "/wp-content/uploads/">
    <FilesMatch "\.php$">
        Require all denied
    </FilesMatch>
</Directory>
```

Das ist die **wichtigste Einzelmaßnahme** überhaupt, weil viele Malware-Uploads/Backdoors genau darauf abzielen, eine PHP-Datei in `uploads/` abzulegen und dann per Direktaufruf auszuführen. `uploads/` sollte generell **niemals PHP ausführen dürfen**.

Was FlyWP wahrscheinlich konkret macht

FlyWP (und ähnliche Managed-Hosting-Tools wie Kinsta, RunCloud, GridPane) bieten meist vorkonfigurierte Regeln für genau diese Kombination:

1. PHP-Ausführung in `wp-content/uploads` blocken
2. Direkten PHP-Zugriff auf `wp-includes` blocken
3. Directory Listing (Browsing) in beiden Ordnern deaktivieren
4. Zugriff auf sensible Dateien (`readme.html`, `wp-config.php`, `.git`, etc.) sperren

Das ist im Prinzip Standard-Hardening, wie es auch im offiziellen WordPress Hardening Guide empfohlen wird.

Fazit

- **Ja, aktivieren** – es ist ein Low-Risk/High-Value-Hardening-Schritt.
- Wichtig ist nur: **Assets (CSS/JS/Bilder) müssen weiter erreichbar bleiben**, nur die **PHP-Ausführung** in Upload-Verzeichnissen und der **direkte Aufruf von Core-PHP-Dateien** in `wp-includes` sollte blockiert werden.
- Danach kurz testen: Seite laden, Bilder/CSS/JS prüfen, ggf. Page-Builder/Plugin-Funktionen (manche legen z.B. generierte CSS-Dateien in `uploads/` ab – das bleibt unberührt, da nur

PHP-Ausführung gesperrt wird, keine anderen Dateitypen).

Wenn du willst, kann ich dir konkrete .htaccess- oder Nginx-Regeln für deinen Stack zusammenstellen.