WPSecureStack

Web Server Security Rules

wpsecurestack.com

This document contains WordPress security rules for all four major web server types: **Apache, Nginx, LiteSpeed, and IIS.** Each part is self-contained — apply only the section that matches your server. Applying the wrong section will cause errors. Each section covers the same 13 security rules adapted to the correct server syntax.

CONTENTS

HOW TO IDENTIFY YOUR SERVER TYPE

Ask your host	The safest and fastest method. Email or live-chat your hosting provider and ask: 'Which web server software does my hosting plan use?'
Check response headers	Visit https://securityheaders.com or https://whatsmyheadinfo.com and enter your domain. Look for the Server: header — it usually says Apache, nginx, LiteSpeed, or Microsoft-IIS.
Check your control panel	cPanel = almost always Apache or LiteSpeed. Plesk = Apache or Nginx. Kinsta / WP Engine / Cloudflare Pages = Nginx. Windows hosts = IIS.

DISCLAIMER

1. Your Responsibility	You are solely responsible for reviewing, testing, and adapting these rules to your specific server environment before deploying to any live site. Rules that work on one host may behave differently on another. Always test on a staging environment first.
2. Server Type Compatibility	Each section of this document is clearly labelled for one server type only — Apache, Nginx, LiteSpeed, or IIS. Applying rules from the wrong section to your server will cause errors or may take your site offline. If you are unsure which server your host uses, ask them before applying any rules from this document.
3. Always Backup First	A mistake in a server config or .htaccess file can take your entire site offline instantly. Before making any changes, download a copy of your existing config file. WPSecureStack accepts no responsibility for downtime, broken functionality, or data loss caused by applying these rules without a prior backup.
4. Not a Complete Security Solution	These rules are one layer of a multi-layered security strategy. They do not replace a Web Application Firewall (WAF), malware scanner, strong passwords, two-factor authentication, or regular updates. No server config file alone can fully secure a website.
5. Plugin and Theme Conflicts	Caching plugins (WP Rocket, W3 Total Cache), page builders, and WooCommerce add their own rewrite rules and headers. Conflicts between those rules and these rules may cause unexpected behaviour. Review your full config carefully after applying.

6. IP Restriction Warning	Any rules that restrict access by IP address require that you have a static IP. If your IP address changes (as it does on most home broadband connections), you may lock yourself out of your own admin panel. WPSecureStack is not responsible for admin lockouts caused by IP-based rules.
7. Managed Hosting Environments	Kinsta, WP Engine, Flywheel, and similar managed WordPress hosts do not allow direct access to Nginx or server config files. Contact your host's support team — most will apply these rules on request. Do not attempt to add Nginx rules via .htaccess on a managed host.
8. Not Legal or Compliance Advice	Nothing in this document constitutes legal, regulatory, or compliance advice. If your site handles personal data, payment information, or operates in a regulated industry, consult a qualified professional for your compliance obligations (GDPR, PCI-DSS, etc.)
9. No Liability	WPSecureStack, Emmanuel Tatyabala, and any affiliated contributors shall not be held liable for any direct, indirect, incidental, or consequential damages arising from the use or misuse of this document, including but not limited to site downtime, broken functionality, admin lockouts, data loss, or security breaches.

This document is shared in good faith to help site owners improve their security posture. If you are unsure about any rule, contact us before applying it. Email: info@wpsecurestack.com | WhatsApp: +256 785 832 643 | wpsecurestack.com/resources

PART 1 — APACHE

File: /.htaccess in your WordPress root | Applies to: cPanel, shared hosting, most VPS setups with Apache

Before you start: *Download a backup of your existing .htaccess first. Paste these rules ABOVE the # BEGIN WordPress block. Run a quick site test after saving. These rules require Apache with mod_rewrite and mod_headers enabled.*

RULE 1 — PROTECT wp-config.php

Your wp-config.php file contains your database credentials and secret keys. No visitor or bot should ever be able to open it via a browser URL. This rule returns a 403 Forbidden error to anyone who tries.

```
<Files wp-config.php>
    Order Allow,Deny
    Deny from all
</Files>
```

RULE 2 — PROTECT .htaccess ITSELF

The .htaccess file controls your server config. Exposing it publicly lets attackers read your security rules and look for gaps to exploit.

```
<Files .htaccess>
    Order Allow,Deny
    Deny from all
</Files>
```

RULE 3 — BLOCK XML-RPC

xmlrpc.php is a legacy WordPress feature. Attackers use it to run brute-force login attempts and DDoS attacks. Most modern sites do not need it.

Note: *Keep this enabled (comment it out) if you use JetPack or the WordPress mobile app.*

```
<Files xmlrpc.php>
    Order Allow,Deny
    Deny from all
</Files>
```

RULE 4 — BLOCK PHP EXECUTION IN UPLOADS

The uploads folder should only contain images and media — never PHP scripts. This rule stops attackers from uploading and running malicious PHP files.

Note: *Save this rule in /wp-content/uploads/.htaccess — NOT the root .htaccess.*

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteRule \.php$ - [F]
</IfModule>

<Files *.php>
    Order Allow,Deny
    Deny from all
</Files>
```

RULE 5 — BLOCK DIRECT PHP ACCESS IN wp-includes

No file in the wp-includes folder should ever be loaded directly by a browser. This closes a common backdoor exploitation path.

Note: *Multisite users: add 'RewriteCond %{REQUEST_URI} !ms-files\.php' before the RewriteRule.*

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteBase /
    RewriteRule ^wp-includes/[^/]+\.php$ - [F,L]
    RewriteRule ^wp-includes/js/tinymce/langs/.+\.php - [F,L]
    RewriteRule ^wp-includes/theme-compat/ - [F,L]
</IfModule>
```

RULE 6 — DISABLE DIRECTORY BROWSING

Without this, visiting a folder URL may list all files inside — giving attackers a full map of your plugins, themes, and uploaded files.

```
Options -Indexes
```

RULE 7 — WP-ADMIN IP WHITELIST (disabled by default)

Restricts your admin panel to specific IP addresses only. Anyone else gets a 403 error.

Note: *Only enable this if you have a static IP. Replace example IPs before uncommenting.*

```
# <IfModule mod_rewrite.c>
#     RewriteEngine On
#     RewriteCond %{REQUEST_URI} ^/wp-admin [NC]
#     RewriteCond %{REMOTE_ADDR} !^192\.168\.1\.100$
#     RewriteCond %{REMOTE_ADDR} !^203\.0\.113\.50$
#     RewriteRule ^ - [F]
# </IfModule>
```

RULE 8 — BLOCK SENSITIVE FILE TYPES

Prevents log files, SQL dumps, backups, .env files, and archives from being downloaded via a browser URL. A publicly accessible database backup is a critical risk.

```
<FilesMatch "\.(log|sql|bak|backup|gz|tar|zip|env|ini|sh|bash)$">
    Order Allow,Deny
    Deny from all
</FilesMatch>
```

RULE 9 — HIDE WORDPRESS VERSION FILES

readme.html and license.txt reveal which version of WordPress you are running. Attackers use version numbers to look up known vulnerabilities for that exact release.

```
<FilesMatch "^(readme\.html|license\.txt|wp-config-sample\.php)$">
    Order Allow,Deny
    Deny from all
</FilesMatch>
```

RULE 10 — BLOCK DEBUG LOG

If WP_DEBUG_LOG is on and the log is inside wp-content/, this prevents anyone reading it via browser. The debug log can contain file paths and database details.

```
<Files debug.log>
    Order Allow,Deny
    Deny from all
</Files>
```

RULE 11 — BLOCK SCRIPT INJECTION VIA QUERY STRINGS

Blocks common SQL injection and XSS attack patterns that arrive via URL parameters.

Note: *Test this carefully — some plugins use query strings that may trigger false positives.*

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{QUERY_STRING} (<|%3C).*script.*(>|%3E) [NC,OR]
    RewriteCond %{QUERY_STRING} GLOBALS(=|\[|\%[0-9A-Z]{0,2}) [OR]
    RewriteCond %{QUERY_STRING} _REQUEST(=|\[|\%[0-9A-Z]{0,2}) [OR]
    RewriteCond %{QUERY_STRING} (\.\./|%2e%2e%2f|%252e%252e) [OR]
    RewriteCond %{QUERY_STRING} (union|select|insert|drop|delete|update) [NC]
    RewriteRule ^ - [F]
</IfModule>
```

RULE 12 — ADD SECURITY HEADERS

These HTTP headers tell browsers how to behave when loading your site. They protect against clickjacking, MIME sniffing, and other browser-level attacks.

Note: *Content-Security-Policy is commented out — a wrong CSP will break your site's scripts. Only enable it after testing thoroughly.*

```
<IfModule mod_headers.c>
    Header always set X-Frame-Options "SAMEORIGIN"
    Header always set X-Content-Type-Options "nosniff"
    Header always set X-XSS-Protection "1; mode=block"
    Header always set Referrer-Policy "strict-origin-when-cross-origin"
    Header always set Permissions-Policy "geolocation=(), microphone=(), camera=()"
    Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"
    # Uncomment once tested — customise domains to match your setup:
    # Header always set Content-Security-Policy "default-src 'self';"
</IfModule>
```

RULE 13 — LIMIT FILE UPLOAD SIZE

Caps the maximum file size that can be uploaded. Adjust values to suit your site needs.

```
<IfModule mod_php.c>
    php_value upload_max_filesize  64M
    php_value post_max_size        64M
    php_value max_execution_time   300
    php_value max_input_time       300
</IfModule>
```

Apache Pre-Deployment Checklist

[ ]	Backup of existing .htaccess downloaded before making changes
[ ]	Rules pasted ABOVE the # BEGIN WordPress block
[ ]	Rule 4 (PHP in uploads) saved in wp-content/uploads/.htaccess — not root
[ ]	Rule 7 IP whitelist remains commented out unless you have a static IP
[ ]	Site tested in browser immediately after saving — homepage, wp-admin, and image uploads
[ ]	Security headers verified at https://securityheaders.com

PART 2 — NGINX

File: Your server block config (e.g. /etc/nginx/sites-available/yourdomain.com) | Applies to: Kinsta, WP Engine, Cloudways (Nginx), self-managed VPS

Before you start: *Always run* ***nginx -t*** *to test for syntax errors before reloading. Then run* ***systemctl reload nginx*** *to apply. Never skip the test step — a syntax error will crash the server. Paste these rules inside your server { } block.* ***Managed hosts (Kinsta, WP Engine):*** *You cannot edit Nginx directly — contact your host support team and ask them to apply these rules.*

RULE 1 — PROTECT wp-config.php

Blocks all direct browser access to wp-config.php and returns a 403 error.

```
location = /wp-config.php {
    deny all;
    return 403;
}
```

RULE 2 — PROTECT SENSITIVE DOT FILES

Nginx does not use .htaccess, but your server may have .env or .git folders. This blocks all dot files and directories from public access.

```
location ~ /\. {
    deny all;
    return 403;
}
```

RULE 3 — BLOCK XML-RPC

Blocks all access to xmlrpc.php and suppresses log noise from bot traffic.

Note: *Remove this block if you use JetPack or the WordPress mobile app.*

```
location = /xmlrpc.php {
    deny all;
    return 403;
    access_log off;
    log_not_found off;
}
```

RULE 4 — BLOCK PHP EXECUTION IN UPLOADS

Prevents PHP files from being executed inside the uploads folder. Attackers who upload a disguised PHP file cannot run it.

```
location ~* /wp-content/uploads/.*\.php$ {
    deny all;
    return 403;
}
```

RULE 5 — BLOCK DIRECT PHP ACCESS IN wp-includes

Stops direct browser access to PHP files inside wp-includes. Allows the one exception WordPress needs internally.

Note: *Multisite users: also allow ms-files.php — add a separate location block for it.*

```
location ~* ^/wp-includes/[^/]+\.php$ {
    deny all;
    return 403;
}

# Allow this one internal exception WordPress needs:
location ~* ^/wp-includes/ms-files\.php$ {
    allow all;
}
```

RULE 6 — DISABLE DIRECTORY BROWSING

Stops Nginx from listing folder contents when no index file is present. In Nginx, autoindex is off by default — this confirms it explicitly.

```
autoindex off;
```

RULE 7 — WP-ADMIN IP WHITELIST (disabled by default)

Restricts the admin panel to specific IP addresses. All others get a 403 error.

Note: *Only enable if you have a static IP. Replace example IPs before uncommenting.*

```
# location ^~ /wp-admin {
#     allow 192.168.1.100;   # <- REPLACE with your IP
#     allow 203.0.113.50;    # <- REPLACE or remove
#     deny all;
# }
```

RULE 8 — BLOCK SENSITIVE FILE TYPES

Prevents log files, SQL dumps, backups, .env files, and archives from being downloaded via a browser URL.

```
location ~* \.(log|sql|bak|backup|gz|tar|zip|env|ini|sh|bash)$ {
    deny all;
    return 403;
    access_log off;
    log_not_found off;
}
```

RULE 9 — HIDE WORDPRESS VERSION FILES

Blocks direct access to readme.html, license.txt, and the sample config file which reveal your WordPress version number to attackers.

```
location ~* ^/(readme\.html|license\.txt|wp-config-sample\.php)$ {
    deny all;
    return 403;
    access_log off;
    log_not_found off;
}
```

RULE 10 — BLOCK DEBUG LOG

Prevents the WordPress debug.log file from being read via a browser URL.

```
location ~* /wp-content/debug\.log$ {
    deny all;
    return 403;
    access_log off;
    log_not_found off;
}
```

RULE 11 — BLOCK SCRIPT INJECTION VIA QUERY STRINGS

Blocks common SQL injection and XSS attack patterns in URL query strings.

Note: *Test carefully — some plugins use query strings that may trigger false positives.*

```
set $block_sql_injections 0;

if ($query_string ~* "(<|%3C).*script.*(>|%3E)") { set $block_sql_injections 1; }
if ($query_string ~* "GLOBALS(=|\[|%[0-9A-Z]{0,2})") { set $block_sql_injections 1; }
if ($query_string ~* "_REQUEST(=|\[|%[0-9A-Z]{0,2})") { set $block_sql_injections 1; }
if ($query_string ~* "(\.\./|%2e%2e%2f|%252e%252e)") { set $block_sql_injections 1; }
if ($query_string ~* "(union|select|insert|drop|delete|update)" ) { set $block_sql_injections 1; }

if ($block_sql_injections = 1) { return 403; }
```

RULE 12 — ADD SECURITY HEADERS

These HTTP headers instruct browsers to protect your visitors against clickjacking, MIME sniffing, and other browser-level attacks.

Note: *Content-Security-Policy is commented out — customise and test it before enabling.*

```
add_header X-Frame-Options "SAMEORIGIN" always;
add_header X-Content-Type-Options "nosniff" always;
add_header X-XSS-Protection "1; mode=block" always;
add_header Referrer-Policy "strict-origin-when-cross-origin" always;
add_header Permissions-Policy "geolocation=(), microphone=(), camera=()" always;
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
# Uncomment and customise once tested:
# add_header Content-Security-Policy "default-src 'self';" always;
```

RULE 13 — LIMIT FILE UPLOAD SIZE

Caps the maximum request body size. Set this in your server or http block.

```
# In your server {} or http {} block:
client_max_body_size 64M;

# In your PHP-FPM location block, also add:
fastcgi_read_timeout 300;
```

Nginx Pre-Deployment Checklist

[ ]	Backup of existing Nginx config saved before making changes
[ ]	All rules pasted inside the server { } block
[ ]	nginx -t run to check for syntax errors — zero errors before reloading
[ ]	systemctl reload nginx run to apply changes
[ ]	Rule 7 IP whitelist remains commented out unless you have a static IP
[ ]	Site tested — homepage, wp-admin login, image uploads all working
[ ]	Security headers verified at https://securityheaders.com

PART 3 — LITESPEED

File: /.htaccess in your WordPress root | Applies to: SiteGround, A2 Hosting, Hostinger, any LiteSpeed-based host

Good news: *LiteSpeed reads Apache-style .htaccess files, so most Apache rules work directly. However, LiteSpeed has its own advanced directives using* ***RewriteRule*** *and native LiteSpeed Cache integration. The rules below are optimised for LiteSpeed and include LSCache-specific notes where relevant. Paste these rules ABOVE the # BEGIN WordPress block.*

RULE 1 — PROTECT wp-config.php

Blocks all direct browser access to wp-config.php. LiteSpeed honours Apache-style Files directives natively.

```
<Files wp-config.php>
    Order Allow,Deny
    Deny from all
</Files>
```

RULE 2 — PROTECT .htaccess AND DOT FILES

Blocks access to .htaccess and any other dot files or directories on the server.

```
<Files .htaccess>
    Order Allow,Deny
    Deny from all
</Files>

# Block all other dot files (.env, .git, etc.)
<FilesMatch "^\.">
    Order Allow,Deny
    Deny from all
</FilesMatch>
```

RULE 3 — BLOCK XML-RPC

Blocks xmlrpc.php. LiteSpeed processes this faster than Apache due to its built-in request handling — set access_log off to reduce log noise.

Note: *Keep enabled if you use JetPack or the WordPress mobile app.*

```
<Files xmlrpc.php>
    Order Allow,Deny
    Deny from all
</Files>
```

RULE 4 — BLOCK PHP EXECUTION IN UPLOADS

Prevents PHP execution in the uploads directory. Save this in /wp-content/uploads/.htaccess — LiteSpeed will apply it to that folder only.

Note: *Save in /wp-content/uploads/.htaccess — not the root .htaccess.*

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteRule \.php$ - [F]
</IfModule>

<Files *.php>
    Order Allow,Deny
    Deny from all
</Files>
```

RULE 5 — BLOCK DIRECT PHP ACCESS IN wp-includes

Blocks direct browser requests to PHP files inside wp-includes. LiteSpeed handles these rewrite conditions natively.

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteBase /
    RewriteRule ^wp-includes/[^/]+\.php$ - [F,L]
    RewriteRule ^wp-includes/js/tinymce/langs/.+\.php - [F,L]
    RewriteRule ^wp-includes/theme-compat/ - [F,L]
</IfModule>
```

RULE 6 — DISABLE DIRECTORY BROWSING

Stops LiteSpeed from listing folder contents. LiteSpeed respects the Options directive in the same way as Apache.

```
Options -Indexes
```

RULE 7 — WP-ADMIN IP WHITELIST (disabled by default)

Restricts admin access to specific IP addresses only.

Note: *Only enable if you have a static IP. Replace the example IPs first.*

```
# <IfModule mod_rewrite.c>
#     RewriteEngine On
#     RewriteCond %{REQUEST_URI} ^/wp-admin [NC]
#     RewriteCond %{REMOTE_ADDR} !^192\.168\.1\.100$
#     RewriteCond %{REMOTE_ADDR} !^203\.0\.113\.50$
#     RewriteRule ^ - [F]
# </IfModule>
```

RULE 8 — BLOCK SENSITIVE FILE TYPES

Blocks log files, SQL dumps, archives, and .env files from public download.

```
<FilesMatch "\.(log|sql|bak|backup|gz|tar|zip|env|ini|sh|bash)$">
    Order Allow,Deny
    Deny from all
</FilesMatch>
```

RULE 9 — HIDE WORDPRESS VERSION FILES

Blocks readme.html and license.txt which expose your WordPress version.

```
<FilesMatch "^(readme\.html|license\.txt|wp-config-sample\.php)$">
    Order Allow,Deny
    Deny from all
</FilesMatch>
```

RULE 10 — BLOCK DEBUG LOG

Prevents the WordPress debug log from being read via a browser URL.

```
<Files debug.log>
    Order Allow,Deny
    Deny from all
</Files>
```

RULE 11 — BLOCK SCRIPT INJECTION VIA QUERY STRINGS

Blocks common SQL injection and XSS patterns in URL parameters. LiteSpeed processes these rewrite conditions natively and efficiently.

Note: *Test carefully — some plugins use query strings that may match these patterns.*

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{QUERY_STRING} (<|%3C).*script.*(>|%3E) [NC,OR]
    RewriteCond %{QUERY_STRING} GLOBALS(=|\[|\%[0-9A-Z]{0,2}) [OR]
    RewriteCond %{QUERY_STRING} _REQUEST(=|\[|\%[0-9A-Z]{0,2}) [OR]
    RewriteCond %{QUERY_STRING} (\.\./|%2e%2e%2f|%252e%252e) [OR]
    RewriteCond %{QUERY_STRING} (union|select|insert|drop|delete|update) [NC]
    RewriteRule ^ - [F]
</IfModule>
```

RULE 12 — ADD SECURITY HEADERS

LiteSpeed supports mod_headers directives in .htaccess. These headers protect visitors against clickjacking, MIME sniffing, and XSS attacks.

Note: *If you use LiteSpeed Cache plugin, check its CDN/Headers tab — some headers may already be set there. Avoid duplicating them.*

```
<IfModule mod_headers.c>
    Header always set X-Frame-Options "SAMEORIGIN"
    Header always set X-Content-Type-Options "nosniff"
    Header always set X-XSS-Protection "1; mode=block"
    Header always set Referrer-Policy "strict-origin-when-cross-origin"
    Header always set Permissions-Policy "geolocation=(), microphone=(), camera=()"
    Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"
    # Header always set Content-Security-Policy "default-src 'self';"
</IfModule>
```

RULE 13 — LIMIT FILE UPLOAD SIZE

Sets the maximum upload file size. On LiteSpeed hosting with cPanel, you can also set this in cPanel > PHP Settings as an alternative.

```
<IfModule mod_php.c>
    php_value upload_max_filesize  64M
    php_value post_max_size        64M
    php_value max_execution_time   300
    php_value max_input_time       300
</IfModule>
```

LiteSpeed Pre-Deployment Checklist

[ ]	Backup of existing .htaccess downloaded before making changes
[ ]	Rules pasted ABOVE the # BEGIN WordPress block
[ ]	Rule 4 (PHP in uploads) saved in wp-content/uploads/.htaccess — not root
[ ]	LiteSpeed Cache plugin headers checked to avoid duplicate security header conflicts
[ ]	Rule 7 IP whitelist remains commented out unless you have a static IP
[ ]	Site tested — homepage, wp-admin, forms, and image uploads all working
[ ]	Security headers verified at https://securityheaders.com

PART 4 — IIS (WINDOWS SERVER)

File: /web.config in your WordPress root | Applies to: Windows hosting, Azure App Service, GoDaddy Windows plans

IIS uses web.config — not .htaccess. *Create a web.config file in your WordPress root if one does not exist. If one already exists, add the rules inside the existing <configuration> tag — do not create duplicate tags. IIS requires the* ***URL Rewrite Module*** *to be installed for rewrite rules to work. Ask your host if it is enabled, or install it from Microsoft's IIS download page.*

All IIS rules below belong inside a single web.config file. The full combined file is shown at the end of this section.

RULE 1 — PROTECT wp-config.php

Blocks all HTTP requests to wp-config.php and returns a 404 response (returning 404 instead of 403 avoids confirming the file exists).

```
<security>
  <requestFiltering>
    <denyUrlSequences>
      <add sequence="wp-config.php" />
    </denyUrlSequences>
  </requestFiltering>
</security>
```

RULE 2 — PROTECT web.config AND DOT FILES

IIS automatically protects web.config from being read via browser. This rule adds explicit protection for .env and other dot files.

```
<security>
  <requestFiltering>
    <hiddenSegments>
      <add segment=".env" />
      <add segment=".git" />
    </hiddenSegments>
    <fileExtensions>
      <add fileExtension=".log" allowed="false" />
      <add fileExtension=".sql" allowed="false" />
      <add fileExtension=".bak" allowed="false" />
      <add fileExtension=".backup" allowed="false" />
      <add fileExtension=".env" allowed="false" />
      <add fileExtension=".ini" allowed="false" />
    </fileExtensions>
  </requestFiltering>
</security>
```

RULE 3 — BLOCK XML-RPC

Blocks all requests to xmlrpc.php using IIS URL Rewrite.

Note: *Remove this rule if you use JetPack or the WordPress mobile app.*

```
<rewrite>
  <rules>
    <rule name="Block xmlrpc.php" stopProcessing="true">
      <match url="^xmlrpc\.php$" />
      <action type="CustomResponse" statusCode="403"
              statusReason="Forbidden"
              statusDescription="Access Denied" />
    </rule>
  </rules>
</rewrite>
```

RULE 4 — BLOCK PHP EXECUTION IN UPLOADS

Blocks direct execution of PHP files inside the uploads folder. In IIS, this is done by removing PHP handler mapping for that directory.

Note: *Add this inside a web.config file placed in wp-content/uploads/ — not the root.*

```
<configuration>
  <system.webServer>
    <handlers>
      <remove name="PHP_via_FastCGI" />
    </handlers>
  </system.webServer>
</configuration>
```

RULE 5 — BLOCK DIRECT ACCESS TO wp-includes PHP FILES

Blocks direct browser requests to PHP files inside wp-includes.

```
<rewrite>
  <rules>
    <rule name="Block wp-includes PHP" stopProcessing="true">
      <match url="^wp-includes/[^/]+\.php$" />
      <action type="CustomResponse" statusCode="403"
              statusReason="Forbidden"
              statusDescription="Access Denied" />
    </rule>
  </rules>
</rewrite>
```

RULE 6 — DISABLE DIRECTORY BROWSING

Stops IIS from showing a list of files when a folder URL is visited directly.

```
<directoryBrowse enabled="false" />
```

RULE 7 — WP-ADMIN IP WHITELIST (disabled by default)

Restricts wp-admin access to specific IP addresses only.

Note: *Only enable if you have a static IP. Replace example IPs before enabling.*

```
<!-- Add inside a web.config file in the /wp-admin/ folder only -->
<!--
<security>
  <ipSecurity allowUnlisted="false">
    <add ipAddress="192.168.1.100" allowed="true" />
    <add ipAddress="203.0.113.50"  allowed="true" />
  </ipSecurity>
</security>
-->
```

RULE 8 — BLOCK SENSITIVE FILE TYPES

Blocks log, SQL dump, archive, and environment files from public download.

```
<security>
  <requestFiltering>
    <fileExtensions>
      <add fileExtension=".log"    allowed="false" />
      <add fileExtension=".sql"    allowed="false" />
      <add fileExtension=".bak"    allowed="false" />
      <add fileExtension=".backup" allowed="false" />
      <add fileExtension=".gz"     allowed="false" />
      <add fileExtension=".tar"    allowed="false" />
      <add fileExtension=".zip"    allowed="false" />
      <add fileExtension=".sh"     allowed="false" />
      <add fileExtension=".bash"   allowed="false" />
    </fileExtensions>
  </requestFiltering>
</security>
```

RULE 9 — HIDE WORDPRESS VERSION FILES

Blocks readme.html, license.txt, and the sample config file via URL Rewrite.

```
<rewrite>
  <rules>
    <rule name="Block WP version files" stopProcessing="true">
      <match url="^(readme\.html|license\.txt|wp-config-sample\.php)$" />
      <action type="CustomResponse" statusCode="403"
              statusReason="Forbidden"
              statusDescription="Access Denied" />
    </rule>
  </rules>
</rewrite>
```

RULE 10 — BLOCK DEBUG LOG

Blocks the debug.log file from being read via a browser URL.

```
<security>
  <requestFiltering>
    <denyUrlSequences>
      <add sequence="debug.log" />
    </denyUrlSequences>
  </requestFiltering>
</security>
```

RULE 11 — BLOCK SCRIPT INJECTION VIA QUERY STRINGS

Blocks common SQL injection and XSS patterns arriving in URL query strings.

Note: *Test carefully — some plugins may trigger false positives.*

```
<rewrite>
  <rules>
    <rule name="Block SQL Injection" stopProcessing="true">
      <match url=".*" />
      <conditions logicalGrouping="MatchAny">
        <add input="{QUERY_STRING}" pattern="(&lt;|%3C).*script.*(&gt;|%3E)" />
        <add input="{QUERY_STRING}" pattern="GLOBALS(=|\[|%[0-9A-Z]{0,2})" />
        <add input="{QUERY_STRING}" pattern="_REQUEST(=|\[|%[0-9A-Z]{0,2})" />
        <add input="{QUERY_STRING}" pattern="(\.\./|%2e%2e%2f)" />
        <add input="{QUERY_STRING}"
             pattern="(union|select|insert|drop|delete|update)"
             ignoreCase="true" />
      </conditions>
      <action type="CustomResponse" statusCode="403"
              statusReason="Forbidden"
              statusDescription="Blocked" />
    </rule>
  </rules>
</rewrite>
```

RULE 12 — ADD SECURITY HEADERS

Adds HTTP security headers to all responses. These protect visitors against clickjacking, MIME sniffing, and other browser-level attacks.

Note: *Strict-Transport-Security only works on HTTPS. Remove it if your site does not have SSL.*

```
<httpProtocol>
  <customHeaders>
    <add name="X-Frame-Options"         value="SAMEORIGIN" />
    <add name="X-Content-Type-Options"  value="nosniff" />
    <add name="X-XSS-Protection"        value="1; mode=block" />
    <add name="Referrer-Policy"         value="strict-origin-when-cross-origin" />
    <add name="Permissions-Policy"      value="geolocation=(), microphone=(), camera=()" />
    <add name="Strict-Transport-Security" value="max-age=31536000; includeSubDomains" />
    <!-- Uncomment and customise CSP once tested:
    <add name="Content-Security-Policy" value="default-src 'self';" />
    -->
  </customHeaders>
</httpProtocol>
```

RULE 13 — LIMIT FILE UPLOAD SIZE

Sets the maximum allowed request size in IIS. The value is in bytes — 67108864 bytes equals 64 MB.

```
<security>
  <requestFiltering>
    <!-- 67108864 bytes = 64 MB -->
    <requestLimits maxAllowedContentLength="67108864" />
  </requestFiltering>
</security>
```

COMBINED web.config — COPY THIS COMPLETE FILE

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.webServer>

    <!-- Rule 6: Disable directory browsing -->
    <directoryBrowse enabled="false" />

    <!-- Rule 12: Security headers -->
    <httpProtocol>
      <customHeaders>
        <add name="X-Frame-Options"           value="SAMEORIGIN" />
        <add name="X-Content-Type-Options"    value="nosniff" />
        <add name="X-XSS-Protection"          value="1; mode=block" />
        <add name="Referrer-Policy"           value="strict-origin-when-cross-origin" />
        <add name="Permissions-Policy"        value="geolocation=(), microphone=(), camera=()" />
        <add name="Strict-Transport-Security" value="max-age=31536000; includeSubDomains" />
      </customHeaders>
    </httpProtocol>

    <!-- Rules 3, 5, 9, 11: URL Rewrite rules -->
    <rewrite>
      <rules>
        <rule name="Block xmlrpc.php" stopProcessing="true">
          <match url="^xmlrpc\.php$" />
          <action type="CustomResponse" statusCode="403"
                  statusReason="Forbidden" statusDescription="Access Denied" />
        </rule>
        <rule name="Block wp-includes PHP files" stopProcessing="true">
          <match url="^wp-includes/[^/]+\.php$" />
          <action type="CustomResponse" statusCode="403"
                  statusReason="Forbidden" statusDescription="Access Denied" />
        </rule>
        <rule name="Block WP version files" stopProcessing="true">
          <match url="^(readme\.html|license\.txt|wp-config-sample\.php)$" />
          <action type="CustomResponse" statusCode="403"
                  statusReason="Forbidden" statusDescription="Access Denied" />
        </rule>
        <rule name="Block SQL Injection" stopProcessing="true">
          <match url=".*" />
          <conditions logicalGrouping="MatchAny">
            <add input="{QUERY_STRING}" pattern="GLOBALS(=|\[)" />
            <add input="{QUERY_STRING}" pattern="_REQUEST(=|\[)" />
            <add input="{QUERY_STRING}" pattern="(\.\./|%2e%2e)" />
            <add input="{QUERY_STRING}"
                 pattern="(union|select|insert|drop|delete|update)"
                 ignoreCase="true" />
          </conditions>
          <action type="CustomResponse" statusCode="403"
                  statusReason="Forbidden" statusDescription="Blocked" />
        </rule>
      </rules>
    </rewrite>

    <!-- Rules 1, 2, 8, 10, 13: Request filtering -->
    <security>
      <requestFiltering>
        <denyUrlSequences>
          <add sequence="wp-config.php" />
          <add sequence="debug.log" />
        </denyUrlSequences>
```

```
        <hiddenSegments>
          <add segment=".env" />
          <add segment=".git" />
        </hiddenSegments>
        <fileExtensions>
          <add fileExtension=".log"    allowed="false" />
          <add fileExtension=".sql"    allowed="false" />
          <add fileExtension=".bak"    allowed="false" />
          <add fileExtension=".backup" allowed="false" />
          <add fileExtension=".gz"     allowed="false" />
          <add fileExtension=".tar"    allowed="false" />
          <add fileExtension=".zip"    allowed="false" />
          <add fileExtension=".env"    allowed="false" />
          <add fileExtension=".ini"    allowed="false" />
          <add fileExtension=".sh"     allowed="false" />
        </fileExtensions>
        <!-- Rule 13: 64MB upload limit -->
        <requestLimits maxAllowedContentLength="67108864" />
      </requestFiltering>
    </security>

  </system.webServer>
</configuration>
```

IIS Pre-Deployment Checklist

[ ]	Backup of existing web.config saved before making changes
[ ]	IIS URL Rewrite Module confirmed as installed (ask host if unsure)
[ ]	Rule 4 (PHP in uploads) placed in a web.config inside wp-content/uploads/ only
[ ]	Rule 7 IP whitelist block remains commented out unless you have a static IP
[ ]	Site tested — homepage, wp-admin login, image uploads, and forms all working
[ ]	Security headers verified at https://securityheaders.com
[ ]	IIS application pool recycled after changes to ensure new config is loaded

---

Emmanuel Tatyabala | Co-Founder, WPSecureStack

info@wpsecurestack.com | WhatsApp: +256 785 832 643

wpsecurestack.com/resources