WordPress Site Security Guide & Checklist

by

Doc Sheldon

of

Intrinsic Value SEO

Table of Contents

Foreword

Dear reader – please understand that there is no such thing as a totally secure website. For every fantastic security feature that a developer can come up with, somewhere, there's another developer who's just a little bit smarter, a little bit more motivated or a little bit more experienced, to overcome it.

The best security tactic to keep hackers out is normally one of two methods:

- make entry so difficult that it's beyond the capabilities of most hackers;
- make it so time-consuming that it's not their time and energy.

In reality, it's usually the second method that's most often employed - it's also still the most effective. However, as automated systems continue to evolve, its effectiveness can be expected to diminish.

It's also worth remembering that with a breach can come reporting requirements to authorities or where applicable, users, and failing to take adequate measures to protect users' personal information can leave you vulnerable to hefty fines and possible civil action. Keeping the website secure against breaches is the first step in protecting that personal information.

That said, there are several things we can do to make our WordPress sites more difficult to breach. Plugging the security holes most often exploited is the first step. In the following 37 tips, I hope you'll find a few security holes you never thought of before. It should be a useful checklist to help you secure your WordPress website.

WordPress Site Security Checklist

Establishing and maintaining the security of our WordPress websites is ultimately our own responsibility. Even if we depend on our hosting service or developer to decide what needs to be done and how to best do it, we still need to take responsibility for ensuring it happens. And like any other aspect of our business, it's not something we can do once and forget about – it's an ongoing process.

Fortunately, the implementation of most security fixes is relatively uncomplicated, often involving simply checking a box, inserting a code snippet or editing a line in a file. This WordPress Site Security Guide is intended to help site owners who have only basic editing skills learn to check their own site's security, and in most cases, avoid a potential issue.

First and foremost, the cardinal rule should be to always backup before making any changes. That doesn't mean just your site's database and files when updating the WP core, theme or plugins – it also means any individual files you might be editing, such as the *.htaccess*, *wp-config.php* or *php.ini* files. It's too easy to make a tiny error that could take your site down – having a recent backup of the file can turn a frown into a smile.

Here are over 3 dozen items you should check on your WP site, along with our recommended actions. Some are critical, some are just precautions that will reduce your risk slightly. And if you know your way around your site's backend and your CPanel, you should be able handle most of these fixes with no problem. They'll make your website less vulnerable to hackers or malware injection attacks.

Don't kid yourself into thinking that attackers won't bother with you because you don't handle any sensitive information… most attacks involve sites that don't handle any financial data or personal information at all – attackers use hijacked sites to inject malware, they use the site to forward spam en masse or they redirect users to other sites.

Basic Security

☐ 1. Ensure your WordPress core is up to date.

One of the most important steps in maintaining your website's security is ensuring the WordPress core is up to date. Whenever a vulnerability is discovered, by the time a new

version has been released to patch it, the vulnerability has already been in the public domain long enough to allow hackers to exploit it – and they can work incredibly fast. Just 2 or 3 days out-of-date can be enough to allow them to infiltrate your site…. so just imagine if you're running 2 or 3 versions behind. [1]

WordPress enables you to set up your site for automatic updates, and if you're not going to keep close tabs and act immediately, I highly recommend you take advantage of this feature. Just go to *Dashboard – Updates* to enable auto-update or to update manually. But remember to always backup your database and files *before* updating.

☐ 2. Are automatic WordPress core updates enabled?

Normally, we recommend allowing automatic minor core updates, the exception being on a highly customized site with substantial modifications. Most minor updates are security patches that won't affect your site's operation. Nevertheless, periodic automatic backups, such as those possible with the iThemes BackupBuddy plugin, can help protect you from lost data in the event of a rare problem encountered after a core update.

☐ 3. Ensure your plugins are up to date.

The second most important aspect of securing your site is to ensure your plugins are kept up to date. Hackers are continuously looking for exploits on plugins and just like with the WP core, they move fast when they find one.

Keeping your plugins up to date is an important way to protect yourself against hacking or injection attacks. Conscientious plugin developers will move quickly to patch any vulnerabilities that come to light, but unfortunately, not all plugin devs are conscientious.

For plugins installed from the WordPress plugin repository, it's easy to see the latest update version and install the latest version. For plugins purchased or otherwise acquired elsewhere, you may have to visit the developer's site to get the latest version.

Most plugins, you'll be able to update from the *Dashboard – Updates* area of your site's backend. Just as with the core, it's always advisable to backup your database and files

[1] *WordPress core updates undergo extensive testing before their release, so issues arising from an update are exceptionally rare. Most post-update issues are caused by plugin incompatibility or improperly implemented customization.*

before updating plugins. That way, if you experience a problem, you can easily recover to the point before the update.

☐ 4. Ensure your theme is up to date.

Keeping your themes up to date is arguably the third most important aspect of your website's security, as they can be as prone to vulnerabilities as plugins. Just as with plugins, if you got the theme from the WP repository, you can update it in the *Dashboard – Updates* section of your backend. You can also do so from *Appearance – Themes*. If it's a premium theme, you may have to update manually after downloading the new version from the developer's site. And again, ALWAYS perform a fresh backup of your database and files before updating the WP core, plugins or your theme.

☐ 5. Are you running a child theme?

Many hackers find exploits for specific plugins and themes, then use automated scripts to find sites that are using them, in order to attack the site. Using a child theme can make it more difficult to identify the theme on which your site runs. It won't prevent a sophisticated hacker from identifying your theme, but it will hide it from the majority of automated attacks. Besides, there are other very good reasons for using a child theme, such as preventing the loss of customizations when a theme is updated.

☐ 6. Are there are deactivated plugins installed?

Even when deactivated, plugins with known vulnerabilities can present an opportunity for hacking or injections, so when you're not using a plugin, it's always best to delete it. If it's a paid plugin you don't want to delete, at least move it out of the */wp-content/plugins/* folder.

☐ 7. Have all your plugins been updated in the last few months?

It's not uncommon to see a plugin that was once extremely popular displaying a *Last Updated* date of 2, 3 or more years past. Sure, that *might* indicate the developer made it rock-solid and it contains no vulnerabilities. More often, though, it means it's been abandoned or forgotten.

I recommend that any plugin that hasn't been updated in 6 months or more should be carefully examined before installing it on your site. A little searching on Google should quickly uncover any relevant online discussion of the plugin, possibly saving you some heartache.

If one of your favorite plugins seems to have been abandoned by the developer, you can often find a replacement or two that can accomplish the same thing. Just be wary of installing plugins that haven't been quality-checked by WordPress or by someone who knows what they're looking at.

☐ 8. Ensure all your plugins are tested to be compatible with your version of WP.

Backwards compatibility doesn't always live up to our hopes. Every reputable plugin developer will test his plugin with each new WordPress core version, to ensure there aren't any compatibility issues. Even then, though, it's not unheard of for some problem to crop up or to see a conflict with some other plugin.

Even though testing isn't infallible, the best protection is to ensure all your plugins have been tested with your version of WordPress before installing and activating them.

☐ 9. Do you have any deactivated themes installed?

Vulnerabilities in themes that aren't activated can still be exploited, so don't keep themes you're not using installed on your site. If it's a premium theme, the very least you should do is move it out of the */wp-content/themes/* folder.

You can delete unused themes in your backend *(Appearance – Themes),* in the *CPanel* or via FTP.

Important Considerations

☐ 10. Does the username "admin" exist, with administrator privileges?

By default, a new WordPress installation will have "admin" as a username. You should change this to something unique and not easily guessed. If someone successfully

determines your administrator username, a brute-force attack to find a password is a remarkably quick process.

If your site still has “admin” as your administrator username, create a new account with administration privileges, assign it a unique username and a complex password, then delete the “admin” account.

☐ 11. Does user with “ID 1” and administrator role exist?

An out-of-the-box WP install will normally assign the default admin account ID 1, which can help an attacker gain access to administrative privileges.

To fix this, create a new user account with admin privileges, then delete the old ID 1 account.

☐ 12. Is your database table prefix still set to the default wp_?

A fresh install of WordPress will set the database table prefix to the default *wp_*. This is easily changed during the initial installation, by changing the table prefix in the *config.php* file. If the site is already live, it’s a bit more complex and probably should only be attempted by someone with phpMyAdmin experience, as you could easily take your site down. This is another situation in which it’s important to backup your database and files before you make any changes.

☐ 13. Is your WordPress database password strong?

Most servers are set up so that the database can’t be accessed from outside the local network. But mistakes happen, so don’t depend on that. Assign a strong password for your database. Conventional wisdom says at least 8 characters long, with a combination of numbers, letters and special characters. I advise extending that to at least 12 characters.

During a fresh install, you’ll be given an opportunity to set up your DB password, but you can also easily change it in your *wp-config.php* file, probably around line 28, which will look something like this:

/** MySQL database password */define(‘DB_PASSWORD’,
‘YOUR_NEW_DB_PASSWORD_GOES_HERE’);

☐ 14. Do your security keys and salts have proper values?

Without going into great detail on how security keys and salts work, suffice it to say that they make the passwords used on your site many times safer against brute-force attacks. They provide a complex encryption, based upon the salts and keys. Essentially, they add seemingly random characters into cookies and passwords, based upon a set pattern.

I've seen sites with no keys and salts in place, which means they were storing unencrypted passwords. Obviously, that could be catastrophic! Fortunately, though, WordPress.org has been kind enough to provide a nifty script to generate new strings.

There are eight keys necessary, and the above script will generate a new, unique set you can drop into your *wp-config.php* file (around lines 48 through 55). They should look something like this:

define('AUTH_KEY', 'xRPog|(YD$1h+HgF-?cE/~|OY~K-]!)`Nph4i5bv|#m+y8-Xp*4E9%,K|{Iu&T0P');
define('SECURE_AUTH_KEY', 'u&/TqCymF{q3ljoE9GU].~DH-$/z~2*IXzOH+G]HtT34b *=9Q|TF~AvDwXLv[A');
define('LOGGED_IN_KEY', '7-9a%T_4Cx]h`VUDSo^Y<WfM[*_$PP4tV#MQxVLw.$nre:HEA:,J t{||W(4cOr');
define('NONCE_KEY', 'lpzae?fp;G9w6G#2RdN,8x*t~P+#+lgk-_+]8;_w+!P*GEdrX]+^5&.r:haQ+6!J');
define('AUTH_SALT', '<!4G[w-1RQo(lh9mA^@H9,FiYD$cNXZ9XTMKW{y#R;Eh`YBv(>r]+-GH->|fo$L9');
define('SECURE_AUTH_SALT', 'DBPi*kN]AbRQ(m;d($$:|^4:8Top4{mHcloj%j{J0s3|onz*|Y!-0K]Cl}s h?}P');
define('LOGGED_IN_SALT', 'E|Q-(=$i{HlV|_+~-++Zm@2`EK,i+lJV8L.jiw|C)#CM<bOO4CPLA 9u`BR9/3e ');
define('NONCE_SALT', '<!4G[w-1RQo(lh9mA^@H9,FiYD$cNXZ9XTMKW{y#R;Eh`YBv(>r]+-GH->|fo$L9');

***Warning**: The above strings are ONLY A SAMPLE. DO **NOT** use these strings – generate your own at the above link.*

☐ 15. Is your WordPress installation address the same as the site address?

Another way to make your site safer against automated attacks is to move the WP core files out of the root directory and into a sub-directory. The simple part of that process is changing the WP address in your backend, in the *Options – General* section. But that's only part of the process.

Note: *You'll only be changing the WP address, NOT the site URL.*

First, go to your CPanel and create a new sub-folder in the root directory. Give it some different name, such as *mysite*.

Now go to the *Options – General* section in your backend and change the WP address to reflect the new location where your WP core files will be located, such as *http://www.example.com/mysite.*

When you make that change, your site will be inaccessible until you complete the next steps. Now, return to your CPanel, and check-mark all the WP files and folders in your root directory, with the exception of the *.htacces*s and *index.php* files and move them into your new sub-directory.

Note: *Any non-WP folders should remain where they are, so leave them unchecked.*

Now you'll need to edit your *index.php* file to reflect the new location of your core files. Look for the line that should read something like this:

/** Loads the WordPress Environment and Template */require('./wp-blog-header.php');

and edit it to reflect the new location, such as:

/** Loads the WordPress Environment and Template */require('./mysite/wp-blog-header.php');

That's the final step in changing your WP installation address. You're done. Just don't forget to change your log-in URL. Since your *wp-admin* folder has been moved, your new log-in URL will need to accommodate that. So from now on, you'll log into your backend at www.example.com/mysite/wp-login.php.

Note: *Any reference to "mysite" above should be replaced with whatever subdirectory name you choose.*

You can also watch this handy video tutorial to help you understand the entire process:

☐ 16. Is *wp-config.php* present in the default location?

This won't help you if someone gains access to your server, via FTP or CPanel, but it can help otherwise.

The idea is to simply move your *wp-config.php* file up one level.

For instance, move it from

/home/www/wp-config.php to ***/home/wp-config.php***

or from

/home/www/my-blog/wp-config.php to ***/home/www/wp-config.php***

☐ 17. What PHP version are you running?

Running an older version of PHP can make your site vulnerable to exploits, as well as cause it to run more slowly. Most reputable hosting companies will readily upgrade your site to PHP v.7, upon request. Many hosting companies allow you to change your version yourself, from within the CPanel.

☐ 18. Check the MySQL version.

Running an old version of MySQL can also slow your site down and make it more vulnerable to hacking, as some low-budget hosting companies run old MySQL versions that are no longer maintained.

If you find that your site is running on an old version of MySQL, contact them and request they update you to a new version. If they can't or won't do so, you should consider moving to another host.

More Advanced

☐ 19. Is the full WordPress version info revealed in the page's meta data?

If you're running the most recent version of WordPress, your risk here is less, but if for some reason, you're a version behind, a revealed WordPress version in the meta data is just asking for trouble. To be safe, I recommend always masking your version.

It's an easy process to remove the version, simply by adding this small code snippet in your theme's *functions.php* file:

function remove_version() {
return '';
} add_filter('the_generator', 'remove_version');

☐ 20. Is your readme.html file accessible via HTTP on its default location?

Your site's *readme.html* file also contains the WP version information, so it should be made impossible to read via HTTP as well.

There are a few different ways to do this. You can give the file a unique filename, relocate it to another location or change its permissions via *chmod* so it can't be accessed via HTTP.

☐ 21. Do your server response headers reveal your PHP version?

Regardless of which version of PHP your site runs on, it's unwise to reveal that information to potential hackers.

Depending upon your host, you may have to request that they configure the HTTP server to not show the PHP version. In some instances, however, you can scrub your server response headers by adding this directive to your .htaccess file:

<IfModule mod_headers.c> Header unset X-Powered-By Header unset Server</IfModule>

☐ 22. Is your expose_php directive turned off?

When blocking the ability to reveal your PHP version, you'll also need to make a change to the expose_php script in your *php.ini* file:

Change this line:

expose_php = on to ***expose_php = off***

☐ 23. Does your site display unnecessary information on a failed login attempt?

Out of the box, WordPress will indicate either wrong username or password when a failed login attempt occurs, which makes it much easier for an attacker to find a valid username, after which, a brute-force attack can be employed to find the password.

A simple modification will instead return a message of "wrong username or password", so the attacker won't know which was wrong. Just add this code snippet into your *functions.php* file:

function wrong_login() { return 'Wrong username or password.';}add_filter('login_errors', 'wrong_login');

☐ 24. Is the "anyone can register" option enabled?

This should be disabled, unless it's essential to your site (an open community forum, for example). Once a savvy hacker gains even limited access to your backend, there's a potential for exploitation.

You can change this in the backend, in the *Options – General* section. Just look for "Membership – anyone can register" and un-check it.

☐ 25. Is general debug mode enabled?

There are three issues with having general debug mode enabled - it will:

1. increase your site's response time;
2. bombard your visitors with messages that may confuse them;
3. provide valuable information to potential attackers.

You can enable or disable general debugging mode for WordPress in your *wp-config.php* file – look (around line 84) for a line similar to:

define('WP_DEBUG', true);

You can delete it, comment it out (add # in front of the line) or change it to:

define('WP_DEBUG', false);

Note: *There may be other debugging scripts employed by a plugin, as well.*

☐ 26. Is JavaScript debug mode enabled?

You may also have a JavaScript debug mode enabled. You can do a search for *script_debug* in your *wp-config.php* to see if you do. If you find it present and it's set to *true*, just change it to *false*, just like you did with the wp_debug mode.

define('SCRIPT_DEBUG', true);

Delete it, comment it out (add# in front of the line) or change it to:

define('SCRIPT_DEBUG', false);

☐ 27. Is the display_errors PHP directive turned off?

In order to avoid revealing any debug information to visitors or potential hackers, all PHP errors should be logged in a safe place. Place this code snippet at the end of your *wp-config.php* file, just above the *require once* function:

ini_set('display_errors', 0);

If you still find PHP errors being displayed, add this line to your *.htaccess* file:

php_flag display_errors Off

If you continue to see PHP errors displaying, try disabling your plugins one at a time, to learn which one is enabling error display.

☐ 28. Does the wp-config.php file have the right permissions (chmod) set?

There's a lot of sensitive information contained in the *wp-config.php* file… plain-text information that you definitely don't want falling into the wrong hands.
The *chmod* setting you'll want may vary for your *wp-config.php* file, depending upon how your server is configured, but if it's a Linux server, you'll probably be okay with a *wp_config chmod* setting of either 0400 or 0440. That will afford you a lot more protection than a 0644 setting.

☐ 29. Ensure the install.php file isn't accessible via HTTP in the default location

After your WP installation is complete, you won't need this file again, so there's no sense in letting it remain accessible via HTTP, at least not in its native location.

You'll find the *install.php* file located in the *wp-admin* folder. You can delete it, rename it, move it or *chmod* it to prevent HTTP access… your choice.

☐ 30. Is the upgrade.php file accessible via HTTP in the default location?

Don't delete this file, because you may need it later, but it shouldn't be allowed to be accessible via HTTP in its default location. Rename it or move it – or *chmod* it if you prefer. Just make sure your *upgrade.php* file can't be accessed by HTTP within the *wp-admin* folder.

☐ 31. Is the register_globals PHP directive turned off?

This is one of the highest risk security issues I've seen on a WP site. Any hosting company that has this directive enabled should be kicked to the curb immediately. You can read more about why this is so dangerous on PHP manual.

If you have access to *php.ini* file, locate this line:

register_globals = on

and change it to ***register_globals = off***

If you don't have *php.ini* access, you can add this directive into your *.htaccess* file:

php_flag register_globals off

If that doesn't do the trick, you should contact a security professional.

☐ 32. Is the plugins/themes file editor enabled?

It's really handy being able to edit themes and plugins from the backend file editor, but it also presents a security risk. If an attacker can gain access to your admin account, they can do a lot of injection mischief via that editor.

You can easily disable the file editor by adding this code snippet to the *functions.php* file:

define('DISALLOW_FILE_EDIT', true);

If you want to use the editor, you'll have to temporarily comment out that line.

☐ 33. Is the uploads folder browsable by browsers?

If someone can access your uploads folder via their browser, they can download all your uploaded files, which presents both a security risk and copyright issues.

The problem of having your uploads folder browsable can be easily fixed by adding this directive to the *.htaccess* file:

Options -Indexes

☐ 34. Is PHP safe mode disabled?

Some hosts attempt to overcome the security problems inherent to shared servers via the PHP safe mode. This isn't the right way to do it, which is why it's been deprecated since PHP 5.3. If this is the solution offered by your host, you're probably well-advised to look for a new home for your site.

You can turn it off, provided you have *php.ini* access, by changing:

safe_mode = on to ***safe_mode = off***

☐ 35. Is the allow_url_include PHP directive turned off?

If this PHP directive is enabled, it can leave your site vulnerable to XSS cross-site attacks. There's no benefit whatsoever to having this include enabled – only elevated risk.

If you can access the *php.ini* file, change:

allow_url_include = on to ***allow_url_include = off***

If that doesn't disable the *allow_url_include*, you should contact a security professional.

☐ 36. Is EditURI link present in pages' header data?

There's no need to have the EditURI link in the pages' header data unless you're using RSD (Really Simple Discovery) services, such as pingbacks. To remove it, just add this line to your *functions.php* file:

remove_action('wp_head', 'rsd_link');

To completely disable XML-RPC functions, add this snippet to your *wp-config.php* just below the *require_once(ABSPATH . 'wp-settings.php');* line:

add_filter('xmlrpc_enabled', '__return_false');

Then add this snippet to your *.htaccess* file to prevent DDoS attacks:

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

☐ 37. Is the Windows Live Writer link present in the pages' header data?

Unless you're using Windows Live Writer, you should remove this link from the header. Why tell the whole world you're using WordPress?

It's an easy fix. Just add this snippet to the *functions.php* file:

remove_action('wp_head', 'wlwmanifest_link');

While there's no 100% certain way to keep determined attackers out of any website, these tips will help you minimize the risks to your site from hackers, malware and injection attacks. If you get hung up at any point, find a professional to help you out.

Bear in mind – this isn't an all-inclusive list of everything that can be done to make a WordPress site more secure. It's just a list of suggested actions to protect some items that have proven to be particularly easy to exploit. Think of it like this: locking your front door won't keep a determined burglar from kicking the door in – but adding a deadbolt, security chain and a blocking bar will certainly slow him down… and quite probably convince him to go elsewhere… someplace easier to get into.

If you need help, you can always call us at Intrinsic Value SEO. Building and Securing WordPress sites is one of our specialties.

Intrinsic Value SEO
doc@intrinsicvalueseo.com
+1(619)550-2872