WordPress Security White Paper

Last updated September 2025

Overview

This document is an analysis and explanation of the WordPress core software development and its related security processes, as well as an examination of the inherent security built directly into the software. Decision makers evaluating WordPress as a content management system or web application framework should use this document in their analysis and decision-making, and for developers to refer to it to familiarize themselves with the security components and best practices of the software.

The information in this document is up-to-date for the latest stable release of the software, but should be considered relevant also to more recent versions of the software as backwards compatibility is a strong focus for the WordPress development team. Specific security measures and changes will be noted as they have been added to the core software in specific releases. It is strongly encouraged to always be running the latest stable version of WordPress to ensure the most secure experience possible.

Executive Summary

WordPress is a dynamic open source content management system which is used to power millions of websites, web applications, and blogs. It currently powers more than 43% of the top 10 million websites on the Internet. WordPress' usability, extensibility, and mature development community make it a popular and secure choice for websites of all sizes.

Table of Contents

Since its inception in 2003, WordPress has undergone continual hardening so its core software can address and mitigate common security threats, including the Top 10 list identified by The Open Web Application Security Project (OWASP) as common security vulnerabilities, which are discussed in this document.

The WordPress Security Team, in collaboration with the WordPress Core Leadership Team and backed by the WordPress global community, works to identify and resolve security issues in the core software available for distribution and installation at WordPress.org, as well as recommending and documenting security best practices for third-party plugin and theme authors.

Site developers and administrators should pay particular attention to the correct use of core APIs and underlying server configuration which have been the source of common vulnerabilities, as well as ensuring all users employ strong passwords to access WordPress.

An Overview of WordPress

WordPress is a free and open source content management system (CMS). It is the most widely-used CMS software in the world and it powers more than 43% of the top 10 million websites[1], giving it an estimated 63% market share of all sites using a CMS.

WordPress is licensed under the General Public License (GPLv2 or later) which provides four core freedoms, and can be considered as the WordPress "bill of rights":

1. The freedom to run the program, for any purpose.
2. The freedom to study how the program works, and change it to make it do what you wish.
3. The freedom to redistribute.
4. The freedom to distribute copies of your modified versions to others.

The WordPress Core Leadership Team

The WordPress project is a meritocracy, run by a core leadership team, and led by its co-creator and lead developer, Matt Mullenweg. The team governs all aspects of the project, including core development, WordPress.org, and community initiatives.

The Core Leadership Team consists of Matt Mullenweg, five lead developers, and more than a dozen core developers with permanent commit access. These developers have final authority on technical decisions, and lead architecture discussions and implementation efforts.

WordPress has a number of contributing developers. Some of these are former or current committers, and some are likely future committers. These contributing developers are trusted and veteran contributors to WordPress who have earned a great deal of respect among their peers. As needed, WordPress also has guest committers, individuals who are granted commit access, sometimes for a specific component, on a temporary or trial basis.

The core and contributing developers primarily guide WordPress development. Every version, hundreds of developers contribute code to WordPress. These core contributors are volunteers who contribute to the core codebase in some way.

The WordPress Release Cycle

Each WordPress release cycle is led by one or more of the core WordPress developers. A release cycle usually lasts around 4 months from the initial scoping meeting to launch of the version.

A release cycle follows the following pattern[2]:

- Phase 1: Planning and securing team leads. This is done in the #core chat room on Slack. The release lead discusses features for the next release of WordPress. WordPress contributors get involved with that discussion. The release lead will identify team leads for each of the features.
- Phase 2: Development work begins. Team leads assemble teams and work on their assigned features. Regular chats are scheduled to ensure the development keeps moving forward.
- Phase 3: Beta. Betas are released, and beta-testers are asked to start reporting bugs. No more commits for new enhancements or feature requests are carried out from this phase on. Third-party plugin and theme authors are encouraged to test their code against the upcoming changes.
- Phase 4: Release Candidate. There is a string freeze for translatable strings from this point on. Work is targeted on regressions and blockers only.
- Phase 5: Launch. WordPress version is launched and made available in the WordPress Admin for updates.

Version Numbering and Security Releases

A major WordPress version is dictated by the first two sequences. For example, 6.5 is a major release, as is 6.6, 6.7, or 7.0. There isn't a "WordPress 6" or "WordPress 7" and each major release is referred to by its numbering, e.g., "WordPress 6.9."

Major releases may add new user features and developer APIs. Though typically in the software world, a "major" version means you can break backwards compatibility, WordPress strives to never break backwards compatibility. Backwards compatibility is one of the project's most important philosophies, with the aim of making updates much easier on users and developers alike.

A minor WordPress version is dictated by the third sequence. Version 6.5.1 is a minor release, as is 6.4.2[3]. A minor release is reserved for fixing security vulnerabilities and addressing critical bugs only. Since new versions of WordPress are released so frequently — the aim is every 4-5 months for a major release, and minor releases happen as needed — there is only a need for major and minor releases.

Version Backwards Compatibility

The WordPress project has a strong commitment to backwards compatibility. This commitment means that themes, plugins, and custom code continues to function when WordPress core software is updated, encouraging site owners to keep their WordPress version updated to the latest secure release.

WordPress and Security

The WordPress Security Team

The WordPress Security Team is made up of a group of experts including lead developers and security researchers. The team consults with well-known and trusted security researchers and hosting companies[3].

The WordPress Security Team often collaborates with other security teams to address issues in common dependencies, such as resolving the vulnerability in the PHP XML parser, used by the XML-RPC API that ships with WordPress, in WordPress 3.9.2[4]. This vulnerability resolution was a result of a joint effort by both WordPress and Drupal security teams.

WordPress Security Risks, Process, and History

The WordPress Security Team believes in Responsible Disclosure by alerting the security team immediately of any potential vulnerabilities. Potential security vulnerabilities can be signaled to the Security Team via the WordPress HackerOne[5]. The Security Team communicates amongst itself via a private Slack channel, and works on a walled-off, private Trac for tracking, testing, and fixing bugs and security problems.

Each security report is acknowledged upon receipt, and the team works to verify the vulnerability and determine its severity. If confirmed, the security team then plans for a patch to fix the problem which can be committed to an upcoming release of the WordPress software or it can be pushed as an immediate security release, depending on the severity of the issue.

For an immediate security release, an advisory is published by the Security Team to the WordPress.org News site[6] announcing the release and detailing the changes. Credit for the responsible disclosure of a vulnerability is given in the advisory to encourage and reinforce continued responsible reporting in the future.

Administrators of the WordPress software see a notification on their site dashboard to upgrade when a new release is available, and following the manual upgrade users are redirected to the About WordPress screen which details the changes. If administrators have automatic background updates enabled, they will receive an email after an upgrade has been completed.

Automatic Background Updates for Security Releases

WordPress benefits from automated background updates for all minor releases[7]. The WordPress Security Team can identify, fix, and push out automated security enhancements for WordPress without the site owner needing to do anything on their end, and the security update will install automatically.

When a security update is pushed for the current stable release of WordPress, the core team will also push security updates for all releases that are supported and capable of background updates so these older but still recent versions of WordPress will receive security enhancements.

Individual site owners can opt to remove automatic background updates through a simple change in their configuration file, but keeping the functionality is strongly recommended by the core team, as well as running the latest stable release of WordPress.

2013 OWASP Top 10

The Open Web Application Security Project (OWASP) is an online community dedicated to web application security. The OWASP Top 10 list[8] focuses on identifying the most serious application security risks for a broad array of organizations. The Top 10 items are selected and prioritized in combination with consensus estimates of exploitability, detectability, and impact estimates.

The following sections discuss the APIs, resources, and policies that WordPress uses to strengthen the core software and 3rd party plugins and themes against these potential risks.

A1 - Injection

There is a set of functions and APIs available in WordPress to assist developers in making sure unauthorized code cannot be injected, and help them validate and sanitize data. Best practices and documentation are available[9] on how to use these APIs to protect, validate, or sanitize input and output data in HTML, URLs, HTTP headers, and when interacting with the database and filesystem. Administrators can also further restrict the types of file which can be uploaded via filters.

A2 - Broken Authentication and Session Management

WordPress core software manages user accounts and authentication and details such as the user ID, name, and password are managed on the server-side, as well as the authentication cookies. Passwords are protected in the database using standard salting and stretching techniques. Existing sessions are destroyed upon logout.

A3 - Cross Site Scripting (XSS)

WordPress provides a range of functions which can help ensure that user-supplied data is safe[10]. Trusted users, that is administrators and editors on a single WordPress installation, and network administrators only in WordPress Multisite, can post unfiltered HTML or JavaScript as they need to, such as inside a post or page. Untrusted users and user-submitted content is filtered by default to remove dangerous entities, using the KSES library through the `wp_kses` function.

WordPress' range of security-related functions include:

- `esc_html()`[31], `esc_attr()`[32], `esc_url()`[33], and `esc_js()`[34] for escaping the output of user-supplied or untrusted content according to its context.
- `wp_kses()`[35], `wp_kses_post()`[36], `wp_kses_data()`[37] and related functions for filtering out non-whitelisted elements in HTML output.
- Database driver helper functions such as `wpdb::prepare()`[38], `wpdb::update()`[39], and `wpdb::insert()`[40] which help developers write safe SQL queries that don't require manual, unsafe interpolation.
- A range of `sanitize_*()` functions for sanitizing user input according to the context, for example `sanitize_email()`[41] and `sanitize_html_class()`[42].

Proactive measures are also taken to harden the software to protect against misuse. For example, the function `the_search_query()` is often misused by theme authors who do not escape the function's output for use in HTML. The function's output was proactively changed in WordPress 2.3 to be pre-escaped for security.

A4 - Insecure Direct Object Reference

WordPress often provides direct object reference, such as unique numeric identifiers of user accounts or content available in the URL or form fields. While these identifiers disclose direct system information, WordPress' rich permissions and access control system prevent unauthorized requests.

A5 - Security Misconfiguration

The majority of the WordPress security configuration operations are limited to a single authorized administrator. Default settings for WordPress are continually evaluated at the core team level, and the WordPress core team provides documentation and best practices to tighten security for server configuration for running a WordPress site[11].

A6 - Sensitive Data Exposure

WordPress user account passwords are salted and hashed using bcrypt[12], the industry standard approach for secure password hashing. Bcrypt includes a built-in cost factor that makes it computationally unfeasible for attackers to crack passwords through brute force attacks.

To address bcrypt's 72-byte password length limitation, WordPress implements SHA-384 pre-hashing. Application passwords, password reset keys, and other security tokens use the BLAKE2b[50] algorithm via Sodium for enhanced cryptographic security.

WordPress' permission system is used to control access to private information such as registered users' PII, commenters' email addresses, privately published content, etc. A password strength meter is included in the core software providing additional information to users setting their passwords and hints on increasing strength. WordPress also has an optional configuration setting for enforcing HTTPS access for the administration area.

A7 - Missing Function Level Access Control

WordPress checks for proper authorization and permissions for any function level access requests prior to the action being executed. Access or visualization of administrative URLs, menus, and pages without proper authentication is tightly integrated with the authentication system to prevent access from unauthorized users.

A8 - Cross Site Request Forgery (CSRF)

WordPress uses cryptographic tokens, called nonces[13], to validate intent of action requests from authorized users to protect against potential CSRF threats. WordPress provides an API for the generation of these tokens to create and verify unique and temporary tokens, and the token is limited to a specific user, a specific action, a specific object, and a specific time period, which can be added to forms and URLs as needed. Additionally, all nonces are invalidated upon logout.

A9 - Using Components with Known Vulnerabilities

The WordPress core team closely monitors the few included libraries and frameworks WordPress integrates with for core functionality. In the past the core team has made contributions to several third-party components to make them more secure, such as the update to fix a cross-site vulnerability in TinyMCE in WordPress 3.5.2[14].

If necessary, the core team may decide to fork or replace critical external components, such as when the SWFUpload library was officially replaced by the Plupload library in 3.5.2, and a secure fork of SWFUpload was made available by the security team[15] for those plugins who continued to use SWFUpload in the short-term.

A10 - Unvalidated Redirects and Forwards

WordPress' internal access control and authentication system will protect against attempts to direct users to unwanted destinations or automatic redirects. This functionality is also made available to plugin developers via an API, `wp_safe_redirect()`[16].

REST API Security

The WordPress REST API provides a structured way for applications to interact with sites through HTTP requests. While the REST API enables powerful integrations and modern web applications, it also includes security considerations that must be properly addressed to maintain site security.

Authentication Methods

The REST API supports multiple authentication methods, each with specific security implications:

- Cookie Authentication: The standard authentication method included with WordPress. Requires a valid nonce to prevent CSRF attacks. This method is automatically handled for logged-in users within the WordPress admin interface.
- Application Passwords: Allows users to generate application-specific passwords for REST API access. These passwords provide secure authentication without exposing the user's main login credentials, can be easily revoked if compromised, and are not valid for logging into the main admin area.
- Extensible Authentication: Plugins can implement additional authentication methods such as OAuth, JWT, BasicAuth, or platform-specific tokens.

Authorization and Permissions

Each REST API endpoint includes permission callbacks that verify whether the current user has the necessary capabilities to perform the requested action. This ensures that:

- Anonymous users can only access publicly available data
- Authenticated users can only access data and perform actions according to their assigned role(s) and capabilities
- Administrative functions remain restricted to users with appropriate permissions
- By default, some REST API endpoints expose information about users, posts, and site structure. While this information is non-sensitive when queried by users who are not logged in with appropriate permissions, administrators should understand what data is accessible
- Cross-Origin Resource Sharing (CORS): When accessed from web browsers, the REST API serves appropriate CORS headers to prevent unauthorized cross-domain requests
- Custom endpoints must implement permission checks

Data Validation and Sanitization

Each REST API endpoint implements data validation and sanitization practices:

- Input data is validated against expected formats and constraints using JSON Schema, which minimizes the amount of explicit validation that needs to be performed by each endpoint
- Invalid input data is isolated from the callback functions which handle the endpoint

Threat Management

Threat management for the WordPress project covers two areas, the WordPress software and the WordPress.org infrastructure.

The WordPress Software

As a distributed, open source, volunteer driven software project, the security of the source code needs to be maintained and monitored to protect it against malicious actors and unintentional regressions. Various functionality and processes are in place to help protect against these threats:

- A meritocracy based trust process for core committers
- Coding standards[30] that aid legibility and consistency
- A peer review process for every patch that makes its way into the core software
- Open feeds of every commit that goes into the core software, including WordPress Trac's firehose and timeline
- As mentioned above, an active HackerOne program[5] for responsible disclosure of security issues

In addition, the open source nature of the project is an inherent advantage to its security. While closed source, proprietary software is seen by the eyes of just a few, open source software is seen by the eyes of the world. Security and performance issues are reported and fixed by developers from both within and from outside of the WordPress community, and often by people new to the project who take a keen interest in web application security.

The WordPress.org Infrastructure

Automatic updates to WordPress sites are powered by the WordPress.org platform, therefore the security and resilience of this service is vital. WordPress.org is built on the same infrastructure that powers the WordPress.com service, but remains entirely separate. The WordPress.org platform uses a variety of practices to secure its operations, including:

- Globally distributed data centers
- Custom containerized hosting
- A central team constantly monitoring performance
- Load balancing and fault tolerance
- DDoS monitoring and mitigation
- Security compliance, and performance and reliability monitoring
- Hourly backups at multiple offsite locations
- Brute force login detection

- Strict use of roles for user accounts to minimise unnecessary privileges
- Sandbox environments and stricter account security requirements for core team members

Disaster Recovery

Disaster recovery processes are in place and well tested for the entire WordPress.org platform, supported by the hourly backups at multiple offsite locations.

Disaster recovery is less of a concern for the core software, due to its open source and highly distributed nature. Should the source of WordPress be lost or comporomised, there are clones of the software that are kept by many trusted sources.

Further Security Risks and Concerns

XXE (XML eXternal Entity) processing attacks

When processing XML, WordPress disables the loading of custom XML entities to prevent both External Entity and Entity Expansion attacks. Beyond PHP's core functionality, WordPress does not provide additional secure XML processing API for plugin authors.

SSRF (Server Side Request Forgery) Attacks

Outbound HTTP(S) requests issued by WordPress are filtered to prevent access to loopback and private IP addresses. They are also filtered to restrict requests to certain standard HTTP ports (`80`, `443`, and `8080`). This filtering is done by the `wp_http_validate_url()` function, and takes place automatically when the `wp_safe_remote_get()` and `wp_safe_remote_post()` functions are used.

WordPress Plugin and Theme Security

The Default Theme

WordPress requires a theme to be enabled to render content visible on the front end. The default themes which ship with core WordPress have been vigorously reviewed and tested for security reasons by both the team of theme developers plus the core development team.

The default theme can serve as a starting point for custom theme development, and site developers can create a child theme which includes some customization but falls back on the default theme for most functionality and security. The default theme can be easily removed by an administrator if not needed.

WordPress.org Theme and Plugin Repositories

There are approximately 50,000+ plugins and 4,500+ themes listed on the WordPress.org site. These themes and plugins are submitted for inclusion and are manually reviewed by volunteers before making them available on the repository.

Inclusion of plugins and themes in the repository is not a guarantee that they are free from security vulnerabilities. Guidelines are provided for plugin authors to consult prior to submission for inclusion in the repository[17], and extensive documentation about how to do WordPress theme development[18] is provided on the WordPress.org site.

Each plugin and theme has the ability to be continually developed by the plugin or theme owner, and any subsequent fixes or feature development can be uploaded to the repository and made available to users with that plugin or theme installed with a description of that change. Site administrators are notified of plugins which need to be updated via their administration dashboard.

When a plugin vulnerability is discovered by the WordPress Security Team, they contact the plugin author and work together to fix and release a secure version of the plugin. If there is a lack of response from the plugin author or if the vulnerability is severe, the plugin/theme is pulled from the public directory, and in some cases, fixed and updated directly by the Security Team.

The Theme Review Team

The Theme Review Team is a group of volunteers, led by key and established members of the WordPress community, who review and approve themes submitted to be included in the official WordPress Theme directory. The Theme Review Team maintains the official Theme Review Guidelines[19], the Theme Unit Test Data[20], and the Theme Check Plugin[21], and attempts to engage and educate the WordPress Theme developer community regarding development best practices. Inclusion in the group is moderated by core committers of the WordPress development team.

The Role of the Hosting Provider in WordPress Security

WordPress can be installed on a multitude of platforms. Though WordPress core software provides many provisions for operating a secure web application, which were covered in this document, the configuration of the operating system and the underlying web server hosting the software is equally important to keep the WordPress applications secure.

A Note about WordPress.com and WordPress security

WordPress.com is the largest WordPress installation in the world, and is owned and managed by Automattic, Inc., which was founded by Matt Mullenweg, the WordPress project co-creator. WordPress.com runs on the core WordPress software, and has its own security processes, risks, and solutions[22]. This document refers to security regarding the self-hosted, downloadable open source WordPress software available from WordPress.org and installable on any server in the world.

Appendix

Core WordPress APIs

The WordPress Core Application Programming Interface (API) is comprised of several individual APIs[23], each one covering the functions involved in, and use of, a given set of functionality. Together, these form the project interface which allows plugins and themes to interact with, alter, and extend WordPress core functionality safely and securely.

While each WordPress API provides best practices and standardized ways to interact with and extend WordPress core software, the following WordPress APIs are the most pertinent to enforcing and hardening WordPress security:

Database API

The Database API[24] provides the correct methods for safely accessing data which are stored in the database layer.

Filesystem API

The Filesystem API[25] abstracts out the functionality needed for reading and writing local files to the filesystem to be done securely, on a variety of host types.

It does this through the `WP_Filesystem_Base` class, and several subclasses which implement different ways of connecting to the local filesystem, depending on individual host support. Any theme or plugin that needs to write files locally should do so using the WP_Filesystem family of classes.

HTTP API

The HTTP API[27] standardizes the server-side HTTP requests for WordPress. The API handles cookies, gzip encoding and decoding, chunk decoding, and various other HTTP protocol implementations. The API standardizes requests, tests each method prior to sending, and, based on the server configuration, uses the appropriate method to make the request.

Permissions and current user API

The permissions and current user API[29] is a set of functions which will help verify the current user's permissions and authority to perform any task or operation being requested, and can protect further against unauthorized users accessing or performing functions beyond their permitted capabilities.

WordPress.org Privacy Statement

From time to time, your WordPress site may send data to WordPress.org — including, but not limited to — the version you are using, and a list of installed plugins and themes.

This data is used to provide general enhancements to WordPress, which includes helping to protect your site by finding and automatically installing new updates. It is also used to calculate statistics, such as those shown on the WordPress.org stats page[43].

WordPress.org takes privacy and transparency very seriously. To learn more about what data is collected, and how it is used, please visit the WordPress.org Privacy Policy[44].

White paper content License

Additional Reading

- WordPress News[47]

- WordPress Security releases[48]
- WordPress Developer Resources[49]

Footnotes

- [1] https://w3techs.com/technologies/details/cm-wordpress, as of September 2025.
- [2] https://make.wordpress.org/core/handbook/about/release-cycle/
- [3] Andrew Nacin, WordPress lead developer, https://vip.wordpress.com/security
- [4] https://wordpress.org/news/2014/08/wordpress-3-9-2/
- [5] https://hackerone.com/wordpress
- [6] https://wordpress.org/news/
- [7] https://wordpress.org/news/2013/10/basie/
- [8] https://www.owasp.org/index.php/Top_10_2013-Top_10
- [9] https://developer.wordpress.org/plugins/security/
- [10] https://codex.wordpress.org/Data_Validation#HTML.2FXML
- [11] https://wordpress.org/support/article/hardening-wordpress/
- [12] https://en.wikipedia.org/wiki/Bcrypt
- [13] https://developer.wordpress.org/plugins/security/nonces/
- [14] https://wordpress.org/news/2013/06/wordpress-3-5-2/
- [15] https://make.wordpress.org/core/2013/06/21/secure-swfupload/
- [16] https://developer.wordpress.org/reference/functions/wp_safe_redirect/
- [17] https://wordpress.org/plugins/developers/
- [18] https://developer.wordpress.org/themes/getting-started/
- [19] https://make.wordpress.org/themes/handbook/review/
- [20] https://codex.wordpress.org/Theme_Unit_Test
- [21] https://wordpress.org/plugins/theme-check/
- [22] https://automattic.com/security/
- [23] https://codex.wordpress.org/WordPress_APIs
- [24] https://developer.wordpress.org/apis/handbook/database/
- [25] https://codex.wordpress.org/Filesystem_API
- [27] https://developer.wordpress.org/plugins/http-api/
- [29] https://developer.wordpress.org/reference/functions/current_user_can/
- [30] https://github.com/WordPress-Coding-Standards/WordPress-Coding-Standards
- [31] https://developer.wordpress.org/reference/functions/esc_html/
- [32] https://developer.wordpress.org/reference/functions/esc_attr/
- [33] https://developer.wordpress.org/reference/functions/esc_url/
- [34] https://developer.wordpress.org/reference/functions/esc_js/
- [35] https://developer.wordpress.org/reference/functions/wp_kses/
- [36] https://developer.wordpress.org/reference/functions/wp_kses_post/
- [37] https://developer.wordpress.org/reference/functions/wp_kses_data/
- [38] https://developer.wordpress.org/reference/classes/wpdb/prepare/
- [39] https://developer.wordpress.org/reference/classes/wpdb/update/
- [40] https://developer.wordpress.org/reference/classes/wpdb/insert/
- [41] https://developer.wordpress.org/reference/functions/sanitize_email/
- [42] https://developer.wordpress.org/reference/functions/sanitize_html_class/
- [43] https://wordpress.org/about/stats/

- [44] https://wordpress.org/about/privacy/
- [45] https://wordpressfoundation.org/trademark-policy/
- [46] https://creativecommons.org/publicdomain/zero/1.0/
- [47] https://wordpress.org/news/
- [48] https://wordpress.org/news/category/security/
- [49] https://developer.wordpress.org/
- [50] https://en.wikipedia.org/wiki/BLAKE_(hash_function)