Risk ID	Technical Risk	Technical Risk Indicators	Impact Rating
CWE ID 98	PHP Remote File Inclusion	Application receives user input but does not restrict it before using it in require(), include(), or similar functions	H
CWE ID 89	SQL Injection	Application constructs a dynamic SQL query using a variable derived from user input.	H
CWE ID 259	Hard-coded Password	Method uses a hard coded password,which limits the ability to protect an account.	M
CWE ID 80	Cross-Site Scripting	HTTP responses are populated with user input without validation	M
CWE ID 311	Missing Encryption of Sensitive Data	Application data is sent into a function unencrypted.	M
CWE ID 2327	Use of Broken or Risky Cryptographic Algorithm	Application is using a broken cryptographic algorithm	M
CWE ID 73	External Control of File Name or Path(Directory Traversal)	Argument to a function is a filename constructed using user input.	M
CWE ID 209	Information Exposure Through an Error Message	Error messages may display sensitive information.	L

Impact	Mitigation	Validation Steps
Attacker can specify a URL to a remote location from which the application will retreive code and execute it Locations: wp-admin/update.php 90	Validate all user input to ensure it conforms to an expect format. Use a white list that lists all known safe values.	Attempt to link a url into require that is not on the accepted list, make sure it is not included.
Attacker an execute arbitrary SQL queries against the database, potentially obtaining private information Locations: board.php 30, includes/dblib.php 23, scoreboard/index.php, wp-includes/.../Cache/MySQL.php 344, wp-includes/wp-db.php	Use prepared statements instead of dynamically created SQL queries, to avoid users creating their own malicious querys. In addition, validate user input to make sure it conforms to an expected format.	Make arbitrary SQL queries in all user inputs. Set up alerts to monitor faulty SQL queries being entered.
If one copy of a commercial product is compromised, all may be so. In addition, if the password is compromised, the only way to fix the site is to patch it. Locations: board.php, includes/dblib.php, scoreboard/index.php, wp-admin/.../network/site-new.php	Passwords should be stored away from the application code.	Ensure no hardcoded passwords are used
Attacker can embed malicious input, like Javascript code, into HTTP responses, and they wil lbe executed in the victims browser. They can also steal cookies, modify the presentation of the site, and steal confidential information Locations: board.php, wp-admin/.../ajax-actions.php, wp-admin/async-upload.php, .../includes/class-ftp-pure.php, .../includes/class-ftp-sockets.php , wp-admin/.../class-ftp.php, .../class-phpmailer.php, wp-includes/class-smtp.php, .../class-wp-comments-list-table.php, .../class-wp-editor.php, .../class-wp-list-table.php, .../class-wp-ms-users-list-table.php , wp-admin/edit-comments.php, wp-admin/includes/file.php , wp-admin/import.php, scoreboard/index.php, wp-admin/install.php, wp-admin/link-manager.php, wp-admin/load-scripts.php, wp-admin/load-styles.php, .../media-template.php, wp-admin/includes/media.php, wp-admin/my-sites.php, wp-admin/.../includes/nav-menu.php , wp-admin/plugin-editor.php, wp-admin/plugins.php, wp-admin/includes/post.php, wp-admin/press-this.php, wp-admin/setup-config.php , wp-admin/theme-editor.php, wp-admin/themes.php, wp-admin/widgets.php	Use output filtering to santize all output generated by user-supplied input, to prevent attacks from changing website appearance. In addition, validate user input by using white lists to recognize all acceptable input. Finally, do not permit users to use html content in posts or other data that will be displayed by the application.	Set up flags to be set when users input scripts, make sure input fields do not accept scripts as parameters
Attacker can potentially gain access to data like private cryptographic keys by sniffing the website traffic, .../includes/class-ftp-sockets.php, .../class-wp-filesystem-ftpext.php	Ensure that the application only sends sensitive data as inputs to functions if it is encrypted.	Sniff the network for confidential information that is sent in plaintext, should not discover any.

If an attacker can break the cryptographic algorithm being used, he may be able to gain access to any protected data. Locations: wp-includes/.../Author.php, wp-includes/bookmark.php, wp-includes/.../Caption.php, wp-includes/.../Category.php , wp-admin/.../class-pclzip.php, .../class-phpass.php, .../class-phpmailer.php, wp-admin/.../dashboard.php, wp-includes/cron.php, .../general-template.php, wp-includes/ms-blogs.php , .../ms-functions.php, wp-includes/post.php,	Only use well-tested cryptographic algorithms that are known to not be broken	Static analysis: make sure no calls are made to risky algorithms
An attacker may be able to use this feature to gain unauthorized access to files on the server, including files outside the webroot. Locations: .../includes/class-wp-upgrader.php, wp-includes/.../Engine/shell.php	Validate all user input to ensure it conforms to an expected format.	Attempt to access normally unreachable files through input. Make sure only publicly visible pages are reachable.
For example, by giving a different error message when the username is correct but not the password, you may aid a password cracker. Locations: board.php , includes/dblib.php, scoreboard/index.php, wp-admin/plugins.php, wp-admin/network/themes.php	Use only generic error messages that don't reveal unnecessary information	Try all combinations of correct and incorrect input, make sure error messages are generic