Risk ID	Technical Risk	Technical Risk Indicators	Related CVE, CWE, or OSVDB IDs	Impact Rating	Impact	Mitigation	Validation Steps
0	Code Injection	**eval() is run, which can be injected to run any shell code** www/wp-admin/includes/class-pclzip.php:4063 www/wp-admin/press-this.php:205 www/wp-admin/press-this.php:216 www/wp-includes/class-json.php:22 www/wp-includes/js/json2.js:466 www/wp-includes/js/swfupload/swfupload.js:450	CWE 95	H	Any shell code could be injected and run, potentially leading to manipulation of data, loss of files, and/or breach of private files	Don't run dynamically gnenerated strings as code with eval().	Ensure that injecting code does not run on the server
1	SQL Injection	**SQL queries are generated on non-sanitized user input.** www/board.php:30	CWE 89	H	An attacker could run arbitrary SQL queries to view and alter database entries	Properly escape and sanitize user input and ensure it conforms to the expected format	Ensure injections such as a' or '1' = 1' do not successful authenticate users
2	Use of Hard-Coded Password	**Password is compared to hard-coded *Wh@t3ver!Wh@t3ver!*** www/board.php:15	CWE 259	M	Comparing to hard-coded passsword increases the possibility that the account being protected will be compromised; in the possibility of a compromise, all deployed instances are vulnerable; password cannot be changed without patching and rebuilding the software	Store password outside the application code	Ensure passwords are compared as hashes or another encryption algorithm, not hard-coded
3	Cross Site Scripting	**Users' blog posts are posted to public blog without sanitization** www/board.php:43 www/board.php:44 www/board.php:58 www/board.php:59 www/board.php:64	CWE 80	H	Users can directly control (without limit) what HTML is displayed and what Javascript is executed on the blog. This can cause any visitor to the blog to be victims of CSRF, view third-party content under the impression it is from the website creators, be redirected, or be victims of any other Javascript-based attack	Sanitize users' input by eliminating javascript tags, including <script> and <onload>. Raw text and HTML/URL encoded text should be sanitized.	Ensure JavaScript posted in blog posts in <script> tags are not executed in blog page
4	Information Exposure Through an Error Message	**Specific MySQL error messages are displayed to the user** www/board.php:18	CWE 209	M	Attackers can intentionally cause database requests cause errors, and the displayed error messages will give insight to what database software is being used and how the data is organized - this specific information can be used to determine vulnerabilities in the system	Show the user a generic error message that does not show specifics about the implementation	Ensure that failed database queries do not return database-specific information
5	Cookie Tampering	**Cookies are stored as raw plaintext**	CWE 784	L	Attackers can view the values of cookies and easily change them while sending requests using a proxy. This can cause the server to give data to the attacker that is not intended for him/her.	Encrypting cookies so that cookies can be written/read within the server, but not by outside attacks who do not possess the encryption key.	Ensure the cookies are encrypted and not stored as plaintext values