	Technical Risk	Technical Risk Indicators	CWE ID	Impact Rating	Impact	Mitigation	Validation Steps
1	Cross-Site Scripting	Occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script. ((board.php lines 43)	CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page	Medium	Change content of the web application and disrupt access. Execute unauthorized code.	Sanitize user input, use white lists to ensure that it conforms to the expected format, do not permit users to include HTML content in data is displayed by the application.	Try posting data (notes, comments etc.) with script tags in them.
3	SQL Injection	Malicious code is inserted into an entry field for execution in a poorly-designed application and then passed to the backend database. (board.php lines 26-28)	CWE-89: Improper Neutralization of Special Elements used in an SQL Command	High	Gain access to restricted files including the database; loss of confidentiality and integrity of the information.	Input validation and sanitation. Reject any input that does not strictly conform to specifications.	Check to see if user input can query the database.
4	Directory Traversal	The software allows user input to specify paths and file names, and therefore traversing within the system.	CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	Medium	Allows access to directories and files by unauthorized parties, loss of confidentiality.	Validate all user-supplied input to ensure that it conforms to the expected format,	Alter the URL to see if it allows access to other directories.
5	Eval Injection	The software receives input from an upstream component, but it does not neutralize or incorrectly neutralizes code syntax before using the input in a dynamic evaluation call (e.g. “eval"). (index-evilhomer.php line 10)	CWE-95: Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')	Very High	Allows attacker to execute arbitrary code or modify what is executed.	Change the source code to avoid using eval().	Check to see if a system call gets executed through injection.
6	Cookie tampering	Changed ‘admin’ value of cookie from false to true which allowed access access to information that should not have been available.	CWE-565: Reliance on Cookies without Validation and Integrity Checking	High	Gain unauthorized access and privileges, assume false identity.	Avoid storing sensitive information in cookies and add integrity checks to avoid tampering.	Check if tampering with cookies reveals any sensitive information.

7	Credentials Management	Passwords were hardcoded into the application code in cleartext. (board.php line 15)	CWE-259: Use of Hard-coded Password	Medium	Significantly increases the risk of the password being compromised and the password cannot be changed without patching the software.	Store passwords out-of-band from the application code.	Check to see if source code contains password and username in cleartext.
8	User authentication to the WordPress blog can be brute-forced.	The weak password can be cracked using a dictionary attack to reveal that its “football”.	CWE-521: Weak Password Requirements	High	Increased load to the login server which results in slower performance; vulnerability to denial service attacks.	.Lock out user account on 5 incorrect password tries by setting account lockout flag to true.	Account lockout flag set for user account on 5 incorrect password tries.