Risk ID	Technical Risk	Technical Risk Indicators	Impact Rating	Impact	Mitigation	Validation Steps
1	SQL Injection	Modifications to tables or rows in database	H	Leakage or loss of sensitive information	Escape all special characters like quote marks, or use a whitelist of accepted characters for input	Try to inject SQL code into all input fields (like 1' or '1'='1) to ensure code is not being executed.
2	OS Command Injection	Suspiscious commands in server log	H	Virtual shell access allows intruder to do almost anything	Ensure user input is never evaluated	If a user can execute PHP code somewhere, ensure that commands like system() don't work
3	Eval Injection	Suspiscious requests in server log	H	Intruders may execute arbitrary PHP code, giving direct access to the database and OS commands	Avoid risky functions like PHP's eval() or make sure all user input is sanitized before evaluation	Ensure that PHP code is not executed anywhere that user can input text (including query paramaters)
4	Cross-Site Scripting	Insertion of unexpected redirects or other changes made to webpages	M	Loss of control over the webpages that get displayed to your users	Escape angle bracket characters (for tags) in user input	Try inputting a simple script like <script>alert('Hi');</script> into all input fields and ensure the script is never run
5	Credential Management	Unexpected logins to database or server	M	Access and modification to server and/or database	Avoid hard-coding login information and keep all login information outside of the application code	Search through all deployed source code for instances of your login information
6	Information Leakage (Configuration and Errors)	Very specific attacks targeting weaknesses of application	L	Attacker may learn specific information about system, like through phpinfo() function or error messages	Replace specific error messages with generic ones, and prevent user from running functions that give information about system configuration	Intentionally put wrong information into input fields and ensure that error messages are not too revealing (like "This is not the right password for this username")

7	Cookie tampering	Potential unauthorized logins to admin accounts, or just unexpected values in cookies	L	Admin access from unauthorized users	If cookies can be used for login, make sure values are random and hard to guess	Try visiting or logging into pages while modifying simple values in cookies (like integers or booleans)
8	Unsalted Hashes	Developer should know if hashes are not salted	M	Leakage of sensitive information, like user passwords	Salt hashes, or use framework like OAuth that does it for you	See if hashes can be "reversed" using a rainbow or jump table
9	Insecure Crypto Algorithms	Developer should know which crypto algorithms were used	L	Leakage of sensitive information, like user passwords	Use crypto algorithms known to be secure, like SHA-256	Search through database and source code for hashes made with an insecure algorithm
10	Missing Encryption of Sensitive Data	Unexpected logins to database or server	M	Admin access from unauthorized users	Ensure all login information is properly encrypted or not stored within the application	Search application code for plaintext sensitive data
11	Path Traversal	"Touches" to private files or directories by external users	M	Files may be viewed by users that they shouldn't know about	Ensure that arbitrary code cannot be run and that any file operations done by application involving user input uses the proper functions, like realpath(), for doing so	If PHP commands can be run, try using glob() and if user input is used for file access, try using something like ../
12	Execution with Unneccesary Privileges	Deletion or editing of rows or tables in the database	L	Users may make changes to a database or the server itself	Create a new user for the database that can only insert and get rows from one table	Try using default user to delete rows from database console, or even try with SQL injection
13	No Separation of Concerns	Changes to components of the website that users should not have access to	L	Security vulnerability in one part of website leads to a vulnerability in another part	Keep different "concerns" (e.g., application and database) on separate servers	Try to login or access contents of one server from the other