Comp116 Assignment 4 - Technical Risk Analysis - November 13, 2014 - Eric Bailey

ID	Technical Risk	Risk Indicators	Rating	Impact	Mitigation	Validation Steps
1	Code Injection (3 flaws)	Users are able to execute code in the server by putting code as their input	H	Remote code execution on both server and client	Validate user input, do not use eval()	Remove all user-inputted “code” such as SQL queries or <script> tags via regex
2	SQL Injection (7 flaws)	Users are able to mangle SQL queries directly, producing undesired database queries	H	Users are able to log in to accounts that they don’t have access to, and gain information they wouldn’t have normally	Validate user input. Use prepared statements in PHP	Parse through all user input locations (login, comments, usernames, URLs, etc.) and validate input to make sure SQL queries are not messed with
3	Cross-Site Scripting (75 flaws)	Users are running javascript code unintended by the author of this website	M	Users can put in HTML <script> tags to run javascript on other users’ browsers	Validate all user input	Parse through all user input locations (login, comments, usernames, URLs, etc.) and validate input, removing the ability to input “<script>”
4	Cryptographic Issues (100 flaws)	Users are able to see sensitive information in plaintext, passwords not sent encrypted	M	Users can gain access to files and locations that they normally would not be able to	Encrypt all data, Update cryptographic software	Encrypt sensitive data, download the latest updates for encryption libraries, or use different functions
5	Directory Traversals (4 flaws)	Users can access files such as .git by typing it into the URL bar directly	M	Users can navigate to <site>/.git/packed-refs to gain information about the server	Remove .git folder, do not allow users to navigate to *../../*	Do not upload .git folder to server, disallow users from accessing files gotten to by typing something like “../../etc/passwd”
6	Credentials Management (9 flaws)	Hard-coded passwords found in plaintext (“Wh@t3ver!Wh@t3ver!”)	L	Users who can see certain files (via .git, etc.) can gain access	Hash passwords, or do not store them in files on the webserver	Remove all hard-coded passwords from code, replace with queries to files that users do not have access to
7	Information Leakage (7 flaws)	Some source code is available to users when it should not be (keys found in the view-source)	L	Users can find unwanted information via error messages or looking around in the view-source of some files.	Use generic, non-telling error messages that do not give information about back-end insecurities	Simplify error messages to be more generic, reducing the risk of users getting a hint of where to attack. Remove backup or temp files (like .txt~ produced from vim) that may have leaked information to users