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	Н	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</td></tr><tr><td>2</td><td>SQL Injection (7 flaws)</td><td>Users are able to mangle SQL queries directly, producing undesired database queries</td><td>Н</td><td>Users are able to log in to accounts that they don't have access to, and gain information they wouldn't have normally</td><td>Validate user input. Use prepared statements in PHP</td><td>Parse through all user input locations (login, comments, usernames, URLs, etc.) and validate input to make sure SQL queries are not messed with</td></tr><tr><td>3</td><td>Cross-Site Scripting (75 flaws)</td><td>Users are running javascript code unintended by the author of this website</td><td>M</td><td>Users can put in HTML <script> tags to run javascript on other users' browsers</td><td>Validate all user input</td><td>Parse through all user input locations (login, comments, usernames, URLs, etc.) and validate input, removing the ability to input "<script>"</td></tr><tr><td>4</td><td>Cryptographic Issues (100 flaws)</td><td>Users are able to see sensitive information in plaintext, passwords not sent encrypted</td><td>М</td><td>Users can gain access to files and locations that they normally would not be able to</td><td>Encrypt all data, Update cryptographic software</td><td>Encrypt sensitive data, download the latest updates for encryption libraries, or use different functions</td></tr><tr><td>5</td><td>Directory Traversals (4 flaws)</td><td>Users can access files such as .git by typing it into the URL bar directly</td><td>M</td><td>Users can navigate to <site>/.git/packed-refs to gain information about the server</td><td>Remove .git folder, do not allow users to navigate to *//*</td><td>Do not upload .git folder to server, disallow users from accessing files gotten to by typing something like "//etc/passwd"</td></tr><tr><td>6</td><td>Credentials Management (9 flaws)</td><td>Hard-coded passwords found in plaintext ("Wh@t3ver!Wh@t3ver!")</td><td>L</td><td>Users who can see certain files (via .git, etc.) can gain access</td><td>Hash passwords, or do not store them in files on the webserver</td><td>Remove all hard-coded passwords from code, replace with queries to files that users do not have access to</td></tr><tr><td>7</td><td>Information Leakage (7 flaws)</td><td>Some source code is available to users when it should not be (keys found in the view-source)</td><td>L</td><td>Users can find unwanted information via error messages or looking around in the view-source of some files.</td><td>Use generic, non-telling error messages that do not give information about back-end insecurities</td><td>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</td></tr></tbody></table></script>