Danielle Zelin
Technical Risk Analysis

Comp116 – Assignment 4
Due: 11/19/15

Risk ID	Technical Risk	Technical Risk Indicators	Related CVE, CWE, or OSVDB IDs	Impact Rating	Impact	Mitigation	Validation Steps
1	Code injection is the process of injecting input into an application that evaluates and executes it as code.	If the user's input doesn't conform to the expected format.	CWE ID 95, CWE ID 98	VH/H	Code injection can result in data loss, untrustworthy components of an application, as well as a complete takeover of control.	Validate the user's input to make sure that the input conforms to the expected format. Use white lists (list of acceptable inputs) to detect acceptable inputs. In www/board.php on line 18, the username and password are both not validated. This means that code injection can be used since the input can be anything of the attacker's choice. This also can allow SQL injections and cross-site scripting attacks.	After 5 tries of invalid input, lock the user out for the day. Confirm that the data has not been tampered with.
2	SQL Injection is when the user enters untrusted input into an application which dynamically completes an SQL query.	If the user's input doesn't conform to the expected format.	CWE ID 89	H	The hacker can expose the contents of a SQL database and potentially manipulate the data in the database.	Ensure that the input statements do not directly complete the SQL query. Rather, parameterized prepare statements can be used and change according to the users input. Validate the input using white lists to	Use Penetration Testing to find vulnerabilities when performing SQL injections.

						ensure that it matches the correct format.	
3	Credentials management flaws is an improper management of credentials. If data is stored in plaintext or hardcoded, then it makes the application less secure. It is really hard to fix if the code is in production.	If a password is displayed in the source code or stored in the SQL database in plaintext.	CWE ID 259	M	If a password is hardcoded or coded in plaintext, the hacker can log into the application pretending to be any of the active users.	Encrypt the passwords when stored in the database. This includes a salt and using a valid hash function. Do not store the associated salts and users in plaintext. Store the passwords in a location that is not easily accessible (not in the source code that can be accessed on a web browser). Example: On lines 14-17 in www/board.php, the username, password, database, and host are written directly in the code. And also on line 31 and 111 of www/scoreboard/index.php, the password is displayed in the source code.	Have an additional security question or two step authentication. This could involve sending an email or text message with a code to ensure that it is the actual user logging in.

4	Cross-Site scripting is when an attacker uses a web application to send malicious code that is usually in HTML.	If the input contains content that looks like HTML, then that is a big hint that a Cross-Site Scripting attack may be occurring. For example, phrases like “<script>” or “</script>”	CWE ID 259	M	This type of attack can modify the presentation of the information on a web application, tamper with sensitive information, and manipulate cookies.	Sanitize the user’s input to make sure no less than or greater than signs are used. Make sure words like script, and alert are checked to make sure they aren’t HTML tags.	After several tries of inputting malicious code into an input field, the user should be locked out.
5	Cryptography issues is if data is not encrypted properly and accurately, then sensitive information and data can be exposed.	If attackers have access to the data and can see encrypted passwords and information, it is clear that they will try to decrypt the data.	CWE ID 316, CWE ID 331, CWE ID 311, CWE ID 327	M	If the algorithms and keys used to encrypt the information, the user will be able to access that information easily. Therefore, confidential and sensitive information will be exposed including things like credit card information, passwords, and social security	Use complicated and strong hash algorithms to ensure that only people with the algorithm can decrypt it. Select the appropriate type of cryptography depending on the situation and the data being stored. Store keys securely and follow the best practices for storing keys.	Perform Penetration tests to find vulnerabilities in the encrypted data that an attacker can exploit.

					numbers.		
6	Directory Traversal allows users to input strings used in a file system to access other directories and resources that aren't intended to be accessed.	If the user inputs any strings that resemble system commands that would normally traverse directories, this is an indication that the user is trying to access sensitive information.	CWE ID 73	M	The attacker can access directories and files that aren't intended to be accessed. Therefore, the attacker will have access to private information and can use that information in any malicious way they decide. Passwords, usernames, social security numbers, and other information can be stored in these files as well as encryption keys and other confidential information.	Validate all user-supplied input to make sure that it matches the expected format. Sanitize an routines that are performed with a large amount of iterations and get rid of unwanted characters.	Use penetration testing to determine if the information is accessible. If the use attempts to perform this type of attack several times (looking for key strings like ../../ and etc, bin, usr), then lock the user out.
7	An information leak is when sensitive information is leaked. This information can provide attackers with methods of	If the user can view any sensitive information on a web server, then an attack can be performed. There is	CWE ID 209	L	The attacker can get insight on how the application is built, where information is stored, and what	Make sure the application returns generic error messages and the errors that are generated do not expose information about the application in any way.	Use penetration testing to make sure that no information is leaked when an

	performing other attacks. The information can be leaked in source code, directories, log files or backup files, backend error messages, uninitialized memory, etc.	no way to tell if a user is currently viewing this information, however, there are ways to prevent them from having access to it.			errors exist in the code. Although this information will not directly let the attackers complete an attack, it will provide them with enough information to do so via a different type of attack.	Remove all backup files, and traces of any other files. Make sure that the source code doesn't display any sensitive information.	error occurs as well as when the source code is viewed.
8	Untrusted initialization is when the application trusts variables that have been initialized from external sources.	There is no way to tell if a user is currently finding these flaws.	CWE ID 454	N/A	The attacker can use long command line arguments and it would result in random code that's not planned.	Make sure to compartmentalize the application and be careful with data from external sources.	Limit the amount of characters that can be used in any input field on the application to make sure this arbitrary code will not be accessed.