Stefan Dimitrov - Comp116 - Risk Analysis

Risk ID	Technical Risk	Technical Risk Indicators	Impact Rating	Impact	Mitigation	Validation Steps
1	PHP backend is running **`echo`** on strings concatenated from user input data. **`[index.php]`**	*Evil* hackers performed basic XSS attacks that injected JS code into the index page posts, which displayed annoying popup alerts upon landing the page.	H	XSS attacks are possible. Can break the website, cause redirects to malicious/ phishing websites, etc.	Sanitize user input by escaping for the script tag checking for basic XSS attacks before saving to database and escape the contents of any variables containing user provided data before echoing.	Send malicious (or not) JS code through every possible user input form. Check if the code's post condition.
2	A hardcoded password is used to connect to localhost and access the MySQL database **`board.`** **`[dblib.php]`**	There is a password hardcoded in the source code ☺	H	A malicious user can easily discover the hardcoded password, which then can be used. This will be hard to detect and if detected changing the password to prevent further access will modifying the codebase and downtime.	Store the password outside of the code in an encrypted file with restricted permissions.	Change the database password, update the stored password in the file, then check if the website is still able to function correctly.
3	Incorrect login attempts cause a error message containing the MySQL query generated with the credentials to be displayed. **`[dblib.php]`**	The error message appears whenever incorrect login credentials are sent.	L	This reveals the internal MySQL query constructed by the backend upon a login attempt. This reveals information about the structure of the database and can be potentially exploited by an attacker but adds nothing to the user experience.	Do not display the MySQL query inside the error message. Instead provide some other form of login failure notification that doesn't reveal more information than is needed.	Attempt to login with incorrect credentials.

4	An **`eval`** statement is fed user provided data and the argument is not sanitized. This allows for running PHP code sent by malicious users. **`[index.php]`**	*Evil* hackers were able to exploit this to run UNIX commands on the server by passing the PHP code through modifying a URL parameter.	H	This serious risk allows hackers to fingerprint the server (**phpinfo**), browse the filesystem, possibly causing denial of service by launching a resource intensive process (**find** on /), etc.	Remove all **eval** statements and restructure the code around this. Reconsider the architecture of the system. Restrict evals on arguments that are not user provided. If not possible make sure to sanitize the arguments.	Send malicious (or not) PHP code through every possible user input form. Check if this produces any changes.
5	MySQL queries composed from user provided data and passed to **`mysql_fetch_array`** and **`mysql_query`** are not sanitized. **`[index.php]`**	It is possible to bypass the login screen using a simple SQL injection attack: **' or '1'='1** in the password field.	H	SQL injection attacks are possible. Can lead to leaking database data, unauthorized access to login protected sections of the website.	Sanitize user input by escaping user input and checking for basic SQL injection attacks before processing queries.	Try a variety of SQL injection attacks on the login form and other types of user input.
6	Login credentials are sent in the clear	The protocol used when accessing the login form is http and not https.	M	It is possible to sniff the passwords of admins as well as users which diminishes users' trust and can result in loss of data depending on the permissions of the users compromised.	Enable SSL encryption on the connection with the client.	Using a local proxy or a packet sniffer search for any leakage of credentials as clear text.

7	Cookie stores data in clear text	The cookies store data in clear text and this allows for changing session parameters easily. Namely the parameter 'lg' could be changed easily in order to reveal one of the keys.	M	It is a bad idea to store any sort of data in clear text and cookies are exactly this. Since cookies are most often used to store session or identification data, security and privacy are compromised. Tampering this data could allow unauthorized access or impersonation.	Encrypt any data stored into cookies before storing it on the client.	Using a browser's debugging functionality or some other method, check if the cookies stored on a client are in clear text format.
8	Weak/Common user password	The user pheller has the weak password baseball which is amongst the most commonly used passwords and is trivial to crack.	H	Allowing users to use weak passwords makes it easy for attackers to take over an account, which undermines user privacy, perceived security and doesn't make for happy users either. Moreover admin accounts with weak passwords can do damage.	Force the users to set strong passwords during registration by requiring a minimum length, a minimum number of special characters/capital letters/numbers/etc or even check if the password desired is amongst a list of common passwords.	Attempt to register an account with a weak password.
9	Passwords are stored unsalted	The passwords stored in the users database are not accompanied by a salt.	H	Unsalted password hashes make it much easier for attackers to perform brute force password cracking attacks in the event of a database leak. ahem… http://leakedin.org/	Salt password hashes. This will hinder cracking many passwords quickly and will make rainbow tables less useful.	Check if passwords stored in the database have been salted.

10	A username and password are identical for two user accounts one on the server OS and the other on the website hosted.	There is a user on the server with the username ‘pheller’ and password ‘baseball’ and an account with the same username and password also exists on the website under the users table.	M	Since the website is exposed to the interwebz, cracking that account would be much more likely than cracking the local account yet because a password was used twice and for the same username it is equivalent to cracking the local account.	This is a policy issue, but still a vulnerability. Since the losing the local would be much worse than one user getting hacked, identities should be withheld when creating local accounts on the server (also might thwart social engineering).	Make sure no local accounts are identical to website accounts. Acquaint people with the security policy.
11	/www/includes/ index.html contains a redirect to the upper directory.	The index.html can be accessed by manually typing in the address bar of a client’s web browser which would reveal the upper directory on the server. Depending on the server settings this might or might not reveal the files/folders present.	M	Revealing the directory structure of the server filesystem can reveal useful information that can be used to design an attack.	Do not redirect to the upper directory. If a redirect is necessary reference the particular website subdirectory directly.	Make sure no such redirects are happening in the html files.