Technical Risk Analysis : CTF-Fall 2014

Cameron Jackson

Risk ID	Technical Risk	Technical Risk Indicators	Impact Rating	Impact	Mitigation	Validation Steps
1	SQL database can be manipulated through injection into forms at /board.php and /admin.php (code located in dblib.php)	Typing an individual apostrophe into either form gives an error, which indicates the potential for SQL injection to a potential attacker.	H	All data stored in the database can be read, manipulated, and removed by an outside attacker.	Validate and clean user input in the board.php and admin.php pages. Remove all characters and handle bad input. It is also suggested to change the database use from root, so that if an attacker gets through to the database, they will not have full privileges.	Typing an individual apostrophe does not result in an error.
2	Javascript can be inserted into forms at board.php, admin.php, and Not Global Thermonuclear war (index.html)and executed (Cross-Side Scripting)	Entering "<script>alert('XSS');</script>" into the aforementioned forms results in a Javascript alert box.	M	All client side data and logic is vulnerable. This includes cookies, HTML DOM elements, and any client side input validation.	Validate and clean user input in the board.php and admin.php pages, use a whitelist to keep only input you want instead of a blacklist for malicious input. Avoid directly evaluating user input.	Entering "<script>alert('XSS');</script>" into either form does not result in a Javascript alert box.
3	Information from the .git directory can be leaked through directory traversal	Navigation to .git results in a 403 Forbidden, navigation to .git/* downloads the file	M	Gives potential attacker access to configuration files and potentially source code.	Remove the .git directory from the list of files that the server hosts	Navigation to /.git/* results in a 404 Not Found
4	It is possible to gain entry into /wp-admin using brute force	Sending several post requests using wget does not result in a problem; log shows various requests to wp-admin from same IP address	H	Gives attacker access all source code and configuration of file, including database credentials	Lock user out after a set number of password attempts	Sending several post requests to wp-admin results in a lockout.
5	User authentication can be bypassed using cookie tampering	Editing the cookies using a Firefox/Chrome plugin results in user authentication	M	Gives an attacker authentication into a website (and all the permisison that comes with that)	Rely on more than just a cookie to authenticate a user	Editing the cookies does not give authentication.
6	Information about the database is leaked if an error occurs	Editing the database credentials so that they are incorrect gives a error to the user that gives specific details about why the connection failed	L	Gives an attacker more information about your database so that they can mold their next attack	Keep error messages vague so that causing an error does not help an attacker.	A failure to connect gives the user an error that says only that the database failed to connect, not why.
7	Attacker can "piggyback" on another user's authentication and perform tasks	Insert an image tag with the link to logout.php. Clicking on this picture wil log out an authenticated user.	M	Gives an attacker (1) cover to perform dubious deeds and (2) ability to perform actions that require authentication without having to bypass the login form	Attach a request or session ID for authentication-necessary tasks	Clicking an image tage with a link to logout.php does not successfully log out an authenticated user.
8	Lack of cryptographic algorithm for storing passwords allows for easy brute force entry into the admin.php form	Use a plaintext password identifier (or write your own) and note that when a new account is created, the password is stored in plain text	M	Allows easy access to the admin form, decreases the amount of work an attacker will have to do to get into the website.	Encrypt passwords before storing them in the database.	Performing a scan for plaintext passwords does not return positive; cracking the password requires an encryption algorithm or salt.
9	Attacker can send multiple and frequent requests to the server, resulting in a DOS	Using a simple request tool (e.g. curl), sending multiple requests to the server results in noticeable lag time.	H	Can lead to a DOS attack.	Restrict multiple requests from the same IP address within a short amount of time.	Sending multiple requests to the server in a short amount of time restricts/or denies access to the server.

10	Attacker can overload the database by sending extremely large input into the "text" fields when created a new post, writing a comment (board.php), and entering a password (admin.php)	Insertion of multiple absurdly large entries into the forms located at board.php and admin.php results in a noticeable lag time in database retrieval	L	Can slow down overall performance of the site, leading to reduced revenue (in a consumer facing site).	Limit either the size of the input field or the size of the entry in the database tables "posts", "replies", and "users".	Application refuses to accept entries larger than a predetermined size (that will not tax the database).
11	Attack can easily implement URL redirection to an untrusted site	Using either XSS or SQL Injection, an untrusted link can be inserted into board.php, admin.php, login.php, or logout.php, enticing a user to click on it. Additionally, a user can simply post an unsavory link as a comment or post at board.php	M	Can result in stolen user information or distribution of malware.	Control input validation so that XSS and SQL injection cannot be performed; limit user ability to post unsavory links by filtering posts.	XSS and SQL injection are treated using the mitigation mentioned above; posting a known link to malware results in an error.