Inbar Fried - Comp. 116 - Assignment 4

Risk ID	Technical Risk	Technical Risk Indicators	Impact Rating	Impact	Mitigation	Validation Steps
1	Code injection into the application. The foreign code gets evaluated by the server.	Page contents, such as font colors, background images, and other content is changed upon foreign code execution.	H	Any individual can inject code into input locations on the site and cause the code to be evaluated.	Filter all input that is given to the site and restrict code from being executed by reformatting input from its original identity into strings with added characters such as '//' which cannot be executed by functions like include() or eval().	Code injected to site's application forms no longer has effects as it had previously. Input is validated and filtered before it is processed.
2	SQL injection.	Contents of user database is made visible upon injection of SQL commands into input forms in the site.	H	SQL code can be injected into the site's input forms, providing an attacker with all database information, options to delete database elements, options to modify databse elements and provide themselves access to the site, and an option to view confidential information about other users.	Use white-lists to set a predefined list of acceptable inputs, filter out special characters from input forms, set permission restrictions on the databse that require further validation, and reduce all input to its basic representation and thus preventing encapsulated code.	Database information is no longer available upon simple SQL code injection and certain elements require a second input validation.
3	User authentication to system can be brute-forced.	No access control in place to prevent continual login attempts.	H	Load on login server, invalid access to information	Only allow for a set amount of login attempts before locking the system for a time period. For example, 3 attempts per hour.	System login radio buttons are not accessible after system locks.

4	Cookie intercepting and tampering can be performed.	Cookie is openly accessible and modifiable using browser web tools such as Firebug.	H	Contents of cookie can be modified to set login parameters to true and grant access to attackers without actual valid access authorization.	Encrypt the cookies. Also, have the cookies information stored elsewhere and validate the incoming cookie's information against the local information to check for discrepancies.	Encrypted the cookies' information and now they are no longer easily modifiable. The validation by the server on the correctness of the cookies' information prevents modified cookies from being accepted.
5	Source code is accessible through .git folders.	No permission restrictions set in place for the .git folder. Can be accessed by directory traversal in the URL.	H	Unintended publication of site's source code which can lead to the site's logic being exposed and manipulated by attackers.	Set restrictions on all folders and files using commands like 'chmod'. Restrict permissions to all source code to a very specific list of developers.	.git folder is no longer accessible after resetting permissions using 'chmod'.

6	Credentials management is prone to exploitation.	Passwords are hard-coded into the application.	M	Unauthorized access may be granted to individuals who have gained unauthorized access to these hard-coded passwords, which would lead to all instances of the application being vulnerable to the single compromised password. Also, once this vulnerability has been exploited it is difficult to fix because the software needs to be restructured and redistributed.	Do not hard-code passwords into the application, but instead store them in a separate location. Secure this password storage using further validation and encrypt every password using hashes.	Read through the software to ensure that no passwords have been left hard-coded into the application and that all such passwords have been exported to a remote and secure database.
7	Cross-site scripting (XSS).	Injected script are processed by the application and site's content is modified according to the injected script. All users are redirected to a different page upon visiting the site.	M	Site's content is modified for all users based on the injected scripts' contents, cookies are stolen and modified, and valid users are not able to view the content of the site.	Restrict user input to a predefined white list of valid input, filter out all characters and words indicating a script has been injected (such as <script>), and sanitize all output generated from user input.	All user input containing black-listed characters and words such as <script> are stripped and site's original content is unmodified following attempted script injections.

8	Cryptographic issues. Use of a broken or risky cryptographic algorithm.	A cryptographic algorithm that is known to be broken and insecure is implemented in the code. This might be a faulty re-implementation of a known and published algorithm.	M	Sensitive information that is thought to be secured is actually easily exploited by bypassing the broken/poor cryptographic algorithm used in the code.	Use well-known published cryptographic algorithms that have been proven to be correct, effective, and safe. Trying to reimplement a cryptographic algorithm will cause more harm than good and it will eventually be broken, much sooner than well-known published algorithms.	Inserted a well-known, published, and mathematically proven cryptographic algorithm into the code to secure the information that is currently vulnerable. Now it is not possible to access the encrypted information without extensive amounts of time.
9	Cryptographic issues. Sensitive data is being passed around unecnrypted.	Multiple functions within the application get passed sensitive data such as cryptographic keys in unecrypted manners.	M	Sensitive data is exposed in the code and can be accessed by any unauthorized individual with access to the code, both indirectly (for example, through reverse-engineering) and directly (by looking at the source code).	Protect all sensitive information in the application by encrypting it using well-known, published, and proven cryptographic algorithms.	Sensitive information and parameters are no longer passed around to functions without being encrypted first.

10	Directory traversal in the URL reveals extra information. The ../ attack.	Directory traversal in the URL of the application reveals both a flag.txt file and .git folder, both of which are available without further authentication.	M	Files whose permissions are not restricted and that exist in the project directory path or environment can be accessed using any combination of directory traversal inputs. Files such as "../etc/passwd" can be accessed and sensitive information can be revealed.	Set permission restrictions appropriately on all folders and files within the application environment and filter user input to prevent again character sequences such as "../".	All URL input is sanitized and any "../" character sequence in removed from the input. Permissions have been set appropriately on all folders and files.
11	Cryptographic issues. Insufficient entropy relating to the generation of random numbers.	Use of standard random number generators in the applications code.	M	The pseudorandom numbers generated from these unsafe methods can be brute-forced, providing attackers with the starting key to the cryptographic algorithms used.	In place of the current untrsuted-random number generator, place a trusted cryptographic random number generator that is well-known and published.	Inserted a well-known cryptographic random number generator from OpenSSL instead of the unsafe random number generator present before.

12	Information leakage through an error message.	Error message from the application reveals the location of the file from which the error originated.	L	Information about the environment, users, and associated data is revealed publicly in common error messages. The information from these error messages can either be used directly to exploit the application and act as a building block for a more complicated attack. This is evident in login pages that reject usernames and passwords individually rather than as pairs, revealing to the user if either is a valid input.	Return only very generic error messages to the user that reveal nothing about the inner workings of the application or any other information that could be used maliciously. For example, reject usernames and passwords as a pair; do not reject one specifically and in effect validate the other.	All error messages have been changed to only reveal the bare minimum about the failure of a user action without revealing any further information or "over-sharing".

13	External initialization of trusted variables or data stores.	An unbound string copy allows for overly long command-line inputs that result in a buffer overflow and execution of arbitrary code.	L	An attacker may input overly long arguments to the application, causing the input buffer to overflow and any excess input to be executed as code. The executed code can then perform any action it has been set to do such as expose sensitive information or change settings in the application.	The size of data copied from the optarg variable should be limited to a specific size and any other use of functions that do not restrict input length, such as strcpy(), should either be replaced with safer functions, such as strncpy(), or enforce a size limit on user input before calling these functions.	All instances of strcpy() have been changed to strncpy() and the size of data copied from the optarg variable has a size limit.
14	Information leakage on logout screen.	After logging out from a valid user account, sensitive information is displayed on the logout screen.	L	Sensitive information that is presented to all users upon logout should be concealed to prevent attackers from gaining informaiton about the site. If sensitive information needs to be presented, a reauthentication system should be put in place.	Remove any sensitive information from the logout screen and simply perform the action of logging out and returning the user to the login screen without placing any other information on the screen.	Removed all extra text from the logout screen and made the logout page redirect a user to the initial login screen after logging them out.