Renwar Karim

CSC2031 Coursework Lab Report – Renwar Karim

INTRODUCTION

In this report, for the purposes of examining the app I will be examining its current state with regards to functionality, vulnerabilities and security implementation (and lack therefore of). After that I'll outline the security mechanisms I implemented and justify each of these. Finally, I will conclude the report with an evaluation of my experience doing this coursework.

The LotteryWebApp allows users to register an account and then submit draws. They can then check the result of the lottery to see if they win. They can also view playable draws. There is also a lucky dip function to create a random draw for the user. Admins can create winning draws and play the lottery, viewing any winners. They can also see security logs.

Web app security is crucial because it ensures that data is kept confidential and secure. This is important as the data could be sensitive, so it could harm us if that data were to be compromised. Thus, the web app should minimise security flaws to ensure the data is safe. The CIA triad provides 3 key properties which our app should have: Confidentiality, Integrity and Availability. The 4th property we should also ensure the app has, is Authentication. I will aim to fulfil these properties.

Examination of Lottery Web App

In this section I will outline the findings of my testing of the current state of the Lottery Web App with a focus on any security issues and vulnerabilities.

Security Issue / Vulnerability	Location	Proposed Implementation (with explanation)
PLAINTEXT PASSWORD STORAGE - Registered users have their details stored in plaintext so anyone with access to the database can see their details. That the password is stored in plaintext is especially bad for security as anyone that's able to get into the database, would have immediate access to the passwords. That's bad for security as users may re-use passwords between applications so the hacker would be able to access a user password and use it across many applications	'users' table in lottery database	Passwords should be hashed upon creation then stored in the users table. When the password is authenticated, we can just use the hashing algorithm to authenticate the plaintext password that a user enters on the login page and if it matches when hashed with our hashed version then we know it's the same password.
LACK OF MULTI-USER FUNCTIONALITY – Currently every page is visible to whoever uses the website without the need to login and authenticate themselves. This includes the admin page which is clearly not intended functionality as the admin page should only be visible to someone once they login and pass authentication to confirm they are indeed the admin.	Admin, profile, account, login pages.	Users with user role should be able to access profile page and lottery page but not the admin page. Users with Admin role should be able to access admin page but not the lottery and profile pages.
LACK OF INPUT VALIDATION – Registration fields lack any input validation and allow the user to enter whatever they want.	Register page	Input validation that ensures all fields are entered with correct format.
LOGIN PAGE NOT FUNCTIONAL – Users can register but they cannot login so there is effectively zero user functionality.	Login page	Users should be able to login after registering. The login page should allow authenticated users to login.
LACK OF ERROR PAGES – Currently whenever there is an error, there is no error page shown and the user does not know what error occurred.	Invalid (error) pages	Specific error pages should be implemented (as outline in error pages requirement section).
WINNING DRAWS CREATED BY ADMIN ARE SHOWN IN PLAYABLE DRAWS AND VIEWABLE TO ALL USERS – Non-admin users should not be able to see the winning draw	Lottery page	The winning draw should not be visible to non-admin users when they view playable draws.

before the lottery is played and they click 'check result' button		
LACK OF ENCRYPTION IN STORAGE OF DRAWS AND NO USER KEYS TO ENCRYPT/DECRYPT DRAWS	Draws table	Encrypt Draws before storage and decrypt when they need to be retrieved from Database. Users should have a personal key to encrypt, and decrypt draws
LACK OF ACCESS MANAGEMENT – Currently there are placeholders where there is supposed to be current user info	Home, Account, profile pages	In place of the placeholders on pages such as home, profile and account the relevant info of the user should be displayed.
UNLIMITED LOGIN ATTEMPTS – Although the login page isn't functional yet, there is no attempted functionality to limit login attempts	Login page	Login attempts should be limited to 3 as per the Limiting Invalid Logins requirement.
LACK OF SECURITY LOGS	Lottery.log file	Logging as outlined in Logging requirements on Task 2
NO GUARANTEEED SECURE RANDOMNESS - Currently there is no guarantee of the randomness of generated numbers/draws being cryptographically secure.	Lottery/ admin pages	Random numbers should be securely random and match requirements in Task 2 Secure Randomness section.

Security Programming

Security Mechanism Implemented	Reason for implementation
INPUT VALIDATION (WHITELISTED) – Ensuring the user registration form has the requirements outlined in the Input Validation section in the security requirements list It is whitelisted as this is more practical than blacklisted where we would have to predict all the unexpected data we could receive – this is inconceivable for our registration form as there are endless possibilities of improper input.	One objective of input validation is to protect against input vulnerability attacks which could be a very serious threat to the LotteryWebApp. An example of this could be a SQL injection attack. The other reason to have this mechanism is to ensure that the data that is entered in the registration form is correct as the user may have made an error in their input and e.g., might've accidentally put a symbol in their name such as '$', by validating the input and raising an appropriate error message we reduce the chance of input mistakes like these occurring.
ERROR HANDLING – Ensuring that when an error occurs, the user is redirected to a relevant error page which clearly states the HTTP response status code.	This is beneficial for the possibility that a malicious user may intentionally cause errors to tamper with the app and cause harm so by redirecting them to a separate error page we can stop them causing any harm that may have been made possible by the vulnerability caused by an error.
HASHING PASSWORDS (WITH SALTING) – Passwords are hashed (and salted) at the point of registration so that the database never stores the plaintext password	Storing passwords as plaintext raises a security vulnerability in that, if anybody gained access to the database, they would be able to view and steal passwords without any obstacles (such as hashed passwords) making it easier for them to cause harm. Furthermore, users often re-use passwords between different applications so this could be very bad for the user as it would mean that potentially all their online activities could be compromised. Having passwords stored in their hashed form greatly reduces the risk of passwords being compromised.
USE OF CRYPTOGRAPHY TO ENCRYPT/DECRYPT LOTTERY DRAWS – All the lottery draws are encrypted before being stored in the database and they are decrypted before being viewed and checked for winners.	The reason for encrypting lottery draws is that if the draws were to get compromised then the hacker could then have access to all the draws and most importantly, they may even be able to see the winning draw. Encryption does not

	completely protect against this as the hacker could still decrypt the draw given enough time and resources, but it will make it a lot harder for them, so it improves the overall security of the lottery system. In general, it is best to store anything of a personal nature that is integral to the lottery game in a ciphertext form and only decrypt it when its needs to be viewed/checked by the user with the appropriate permission/role.
TWO FACTOR AUTHENTICATION – When registering a valid authenticator pin key must be entered, this was present in the original app. The new addition is to use this to produce a time-based PIN which must be entered as part of the verification process in logging in.	User-created passwords cannot always be relied upon even with the strong password policy we created as passwords can always get leaked and the user may even share their password accidentally as part of being the victim of a phishing attack so there can always be a situation where user accounts are compromised as hackers could login with just the email and password. Two Factor Authentication (2FA) creates an additional layer of security for a hacker to get past. In this app, it's a unique time-based PIN which is produced by the user using their original PIN that they entered upon registering.
LIMITING INVALID LOGINS – Login attempts should be limited to 3, with appropriate error messages and the login form should be hidden after 3 invalid login attempts.	Without this limiting of logins, it would be very easy for hackers to simply brute force attacks the login form and gain access to accounts by attempting all possible combinations of alphanumeric and symbol characters. Now with this mechanism users cannot make more than 3 login attempts so this should block any brute force attack as brute force attacks rely on being able to make a huge amount of login attempts.
ACCESS MANAGEMENT – All placeholder info to be replaced with user info that is relevant only to the logged in user. Users should only be able to have access to the pages that are relevant to them (i.e., if logged out then only access to home, login and register pages. Only admin has access to admin link/page etc.) If user logged in then home page should show their name and student number, if they aren't logged in then it shouldn't show this.	Anonymous users shouldn't be able to view sections of the website that require a logged in user. This is both helpful from a security and functionality viewpoint as it would reduce the visibility of the app to potentially malicious people who haven't registered whilst also preventing anybody breaking the lottery functionality by submitting draws without being logged in. Access management is also important for ensuring that non-admin users do not have access to admin specific functionality. Furthermore, users who aren't logged in shouldn't be able to logout as there is no need for a user who isn't already logged in to logout.
LOGGING USER ACTIVITY – Logs made when users register, log in, log out, make invalid login and access attempts.	Logging provides a history of all security events that occur on the app. These logs then allow us to gain information on an event if we need to (usually if it is suspicious such as one IP address making a huge amount of invalid login/access attempts). We log the most crucial events that relate directly to security as, if we log everything then we will have a big log file that contains unnecessary logs, making the important logs harder to notice which could delay a security response.
ROLE BASED ACCESS CONTROL – Access to functionality in the app is allowed depending on role.	Instead of setting restrictions for each user that is created, we simply give all users with user role access to profile and lottery page while admin role users get access to admin page. This is

	much less time consuming than setting the access for each user as we could have a huge number of users so that could take a long time to do. It also guarantees we do not accidentally give a user inappropriate access privileges which could happen due to human error when assigning access privileges.
SECURITY HEADERS – Protects against common web security vulnerabilities with pre-defined behaviour.	Instead of having the burden of web security entirely in our own hands (which is inefficient as we may not be experts and could have inadequate security knowledge), the security header is used which handles most of the security of our web content for us.
SECURE RANDOMNESS – Randomness can be obtained easily however cryptographically secured randomness must contain 3 properties: Appearing Random, Unpredictable values and cannot be reliably reproduced after being made.	When creating random draws, we must ensure that we have secure randomness as this is a lottery app so we must ensure that the lucky dip function is fair and producing cryptographically secure draws to ensure that all lucky dip draws are random. Without secure randomness we could have (in the extreme) a case where all 'randomly' created draws are identical which is obviously something we must avoid.

Evaluation

Overall, I am pleased with the outcome of this coursework. I believe I have implemented all required security mechanisms and I am confident that they all work as intended. I have found that some of the security mechanisms have flaws and/or could be improved so I will outline those below.

Security Mechanism	Problem and/or Improvement
INPUT VALIDATION	Password input validation could include easy to guess things such as names or emails so these should not be allowed to be part of a password. PIN key should have validation to check if its base32.
LIMITING INVALID LOGINS	Allows malicious users to abuse the mechanism by way of intentionally locking out lots of accounts.
LOGGING	There is room for more logs to be filtered. E.g., Draw creations and deletions.
SECURITY HEADERS	For extra security, a Strict-Transport-Security header could be used to ensure that users access the app through HTTPS protocol.
ENCRYPTION	Currently only draws and passwords are encrypted. This could be extended to user details such as first and last names as this is also sensitive information. (e.g., if user wishes to play lottery anonymously)

There is also room for more security mechanisms to be implemented which I outline below.

Security Mechanism	Justification/Reason
CAPTCHA	Adds extra protection to the app by preventing bots from accessing the app.
NOTIFYING UPON UNRECOGNSIED LOGIN	Makes users aware when there is a unrecognised login (which is more likely to be malicious than a recognised one) and requests that the user confirms it is them trying to log in.
SPOOFING DETECTION	Protects against spoofing by authenticating data and blocking any of the data that is spoofed.

Although I am happy with how I have finished the coursework, I certainly have areas that I wish I did differently and can improve on if I were to do something similar in the future. One such area is testing, as it is not outlined specifically in the coursework tasks, I admit I did ignore testing for too long. It goes without saying that upon implementing a new feature to a project you should test it. When I finally decided to test the app, I found many bugs from mechanism I had implemented long ago making it harder to fix as it was not fresh in my memory. This wasted a lot of time and made it a lot more difficult to fix these bugs.