**Lab Report**

The following report will demonstrate my understanding of the importance of web app security and how I managed the security risks of a lottery web application. The app provided is supposed to take a user’s registration for the site, their login and then allow them to submit six numbers for their lottery draw. Since the app is intended to be used by multiple users there are many security features which need to be added – this will ensure that users’ data is kept secure and is not visible to others when on the site. It is also important that the lottery draws of the app are encrypted to avoid interception by a third party or other similar security breaches. All of this I will be implementing to the existing code of the application using PyCharm professional.

**Examination**

After taking some time to properly examine the lottery web application which I have been given, I have made some conclusions about the security issues that are present in the app’s current state – this being before any modification to the code base.

One of the main security concerns that I have for this app is how user information is stored and presented in the linked SQLite database. Currently, once connected to the lottery database on PyCharm, the user can see some basic information about all of the users and the admin; their ID in the respective table, their first name, their surname and their phone number. The aforementioned details are all mediocre in terms of a potential security breach. However, there are also details in this table which are in danger of seriously breaching user privacy and security standards. Namely, this include showing the full password of every user, as well as the admin. Clearly this puts all of the users registered to the lottery application at great risk of having their data vulnerable to being stolen. To combat this, I will be implementing a feature which will ensure every user’s sensitive data will be hashed, rather than show their full password.

It is important that, for an application like this, there are separate user and admin privileges – the administrator should have a page where they can view a list of all currently registered users, and users should not be able to view the admin screen at any point during their experience on the site. Likewise, when logged in the admin should never be able to view user-specific pages like the ‘account’ page or ‘lottery’ page as these are intended just for logged in users with user privileges. In its current state, this is possible but I will be looking to change this feature indefinitely by adding role based access control, as this will ensure that users are able to view the right pages based on their role on the app.

At the moment, the application handles errors very poorly. Instead of displaying an error message which utilises the same styling as the lottery app, a generic browser-generated page is shown which does not provide much use to the user. This will have to be updated – currently it features in my ‘TODO’ list. I intend to add the relevant error messages to the most common kinds of errors that users may encounter on a web application, those being: error 400 bad request; error 403 page forbidden; error 404 page not found; error 500 internal server error; error 503 service unavailable. This will greatly improve user experience on the site and make it appear substantially more professional and efficient.

Since the program is intended for use by multiple users, I will need to add and create a login page which uses the information provided in the linked database to validate a user when they are logging in. By implementing a login and logout function into the site, it will be able to handle multiple users logged in simultaneously, thereby fitting the specifications of the project. Having multiple users on the site also means that certain functions will have to be fine-tuned so that they only affect the user currently logged in. For example, once a draw is submitted and the lottery is ran, the user has the ability to delete their draw of numbers and, at present, it also deletes the draws for every user. This should not happen in an app that supports multiple users, and so I will be adding a function which makes changes only for the user who is currently logged in at that time.

**Security Programming**

One of the first security mechanisms that I implemented into the lottery web app was input validation. This is usually done at the beginning of an application’s development as it is essentially the frontal form of protection from malicious input validation attacks; hackers are aware of certain combinations of characters or commands which can be passed through a website’s forms to grant them unauthorised access. With the input validation that I have coded for the lottery app, both the registration and login forms have specific requirements for when expected data is entered into their fields. This is known as whitelisting data, whereby only the expected data is passed through and processed – any invalid data entered is disregarded. As previously mentioned, every field in both of the forms boast some kind of input validation; whether that is ensuring passwords contain a digit, lowercase letter, uppercase letter and a special character, or notifying the user that they have supplied an incorrect two-factor authentication token. This helps with providing the utmost security on the front-end of the program by confirming that users are who they say they are, as well as making sure that the data they enter fits the specific requirements of the app.

Something that was touched upon when discussing input validation was two-factor authentication. This is a relatively new security feature (having only been rolled out in 2013 by Microsoft ) which has become a staple in most website’s logging in experience. For the purpose of this application, I have not went to the lengths that some other websites go to with using memorable phrases, an email link confirmation or CAPTCHA. I have, instead, used an external authenticator application known as Authy, implementing a method of further protecting a user’s account from becoming compromised. The app takes the user’s 32-digit PIN key – set up during the registration process – and generates a 6-digit code which resets every 30 seconds on Authy for that user to enter when logging in to the lottery web app. It would be very difficult for an unwanted person to discover a user’s 32-digit PIN key, comprised of numbers and letters, to then be able to use this in conjunction with the Authy app and successfully guess their password to gain entry. I believe this to be a very strong use of two-factor authentication, especially since it makes full use of an external app to provide an added layer of security.

Due to hackers becoming ever more sophisticated, encryption is a widely-used feature in lots of sites which transfer data. In the modern age, it is now possible for hackers to intercept data while it is in transit – this is quite an alarming thought. To combat this, data cryptography tools are used which support the core CIA triad of cyber security: confidentiality, integrity of data and authentication. The most crucial security service which is provided by cryptography, and a part of the triad, is confidentiality. This refers to maintaining a user’s privacy when on the net, preventing their information from being seen by an unauthorised person. Encrypting data involves converting plaintext that is readable by humans to ciphertext, which is a scrambled kind of text string consisting of letters, numbers and special characters in order to ensure no one can interpret that data. When viewing it in a database, for instance. For my lottery application I have added cryptography in the form of symmetric key encryption; this entails assigning every user who registers onto the blog post a draw key which is used for when they are adding their draws to the lottery draws table. The draw key is then taken as a parameter and used to encrypt each draw into a very long string of ciphertext – essentially an unreadable piece of text in place of the original plaintext. This is very useful so that unwanted eyes cannot read and interpret data which is being uploaded onto the site, or intercept the data in transit. However, we do wish to read this text. So a decryption function must also be applied to the program. It is worth noting that the draws contained within the database are encrypted and unreadable, while the draws seen on the web app can be viewed by the user which is currently logged in, provided they have the ‘user’ access role and permissions. The decryption is creating a local copy of the lottery draw which is independent of its database-stored counterpart. This makes data which is encrypted available to view by the right person, while still having the stored data (located in the database) unreadable by third parties.

While on the topic of data being able to be viewed by the right person, another fundamental security feature which I have added to the lottery web app is access management. This allows the backend coder of the application, being myself, to assign different levels of access for people using the app thereby protecting the sensitive data which certain users, or groups of users, should not have permission to see or interact with. For this particular application, I have considered access management for three different groups: anonymous users, who are not logged in; logged in users; the administrator of the web app. All three have very different permissions and access – this links heavily to role based access control, which has also been implemented into the codebase. Essentially, users who are logged in and have been assigned the role of ‘user’ will have different options during their experience on the site. For example, an anonymous user will not be able to interact with the ‘account’, ‘profile’ or even ‘lottery’ page upon loading the site. It is not until they have first registered an account with the lottery web app and then logged in that they will be able to make use of these pages. This not only makes logical sense by hiding pages which would have no data from an anonymous user to be populated with, but it also adds more depth to the security of the application by ensuring that only users who have been authenticated as members of the site can view and use specific functions. The admin is a separate entity to the standard ‘logged in user’ role. While the admin must log in the same as any other user, when they are using the app they have another page tailored just for them: the ‘admin’ page. From here, they are able to view all of the users who are currently registered onto the site, run the lottery and also view a security log. It is important for a web app with features such as the lottery app is able to differentiate between the range of people who may be accessing or viewing the site, hence why I have employed the use of role based access control and access management so that features available to users are relevant to their role in the site.

One way in which the admin of the site is able to stay on top of access attempts to the site is through the ‘security logs’ feature, which has been briefly mentioned previously. Essentially this is a table which displays essential security information; users logging in and out of the website, invalid login attempts and unauthorised access attempts (such as a user trying to access the admin page). In the table is the user’s ID number, their email and what kind of security notification was logged. This convenient feature just allows the admin to monitor activity on the app and make note of any irregularities or points of interest regarding the site’s security.

**Evaluation**

During the course of developing the lottery web application, I encountered several challenges while implementing the security features. One of the main difficulties that I faced was decrypting users’ lottery draws – I spent a number of hours trying to figure out why my draws were showing up as a random combination of 32 characters, rather then the lottery numbers which should have been assigned to a draw. This did, however, confirm to me that I had successfully implemented the encryption part of the app, though, which meant I could use this as a foundation to work from. After going through the encryption part of my code many times, I realised that while both the encryption and decryption functions had been defined as models in that respective file, I had actually misplaced the decrypt function in the lottery file. Where it should have been utilised in all of the interactive features on this page individually, I had coded a ‘for’ loop to decrypt the posts as soon as the page was loaded, rather than when the buttons were clicked. After cutting the loop from the wrong part of the file, I was able to locate where it should have been in the first place. I ran the code after doing this and I saw that the long-winded combination of numbers, digits and special characters had been replaced by 6 lottery numbers ready to be drawn. This problem did cause me to panic for a short while, simply because encryption is such a crucial tool to be utilised in this kind of application, that if it was to function incorrectly then data could be compromised and prone to attacks.

An example of an attack on data is known as a man-in-the-middle (or MITM) attack, where an unwanted party can intercept data that is not being transferred securely in a variety of ways. Due to time constraints imposed on this project, I was not able to add the maximum amount of security to the application. Had there of been more time allocated to work on the project, I would have added some precautionary measures against MITM attacks, namely a HTTPS URL rather than a HTTP one which the lottery web app currently has. The ‘S’ in HTTPS literally stands for ‘secure’, indicating that the website you are visiting is using TLS encryption to encrypt page requests made by the user and in turn defending their data from being intercepted by a MITM attack. As mentioned, if I had been given more time to work on this project, HTTPS would definitely have been something that I would have added in order to improve the user’s experience on the site by making it more secure and safe.

Overall, I believe that the lottery program is very effective – it works just as intended and also has many security features which are seen as common practice in secure web app development. Working on this has given me a great insight into how web applications are made secure and a user’s experience is improved through effective managing of permissions and access to features on a site.