Lab Report Lottery Web App

Introduction

This report will cover the lottery web app that Newcastle University has decided to set up, it will cover how the site/application stands currently from a security standpoint at first glance, without any editing. The report will then lead into various security features that I have implemented to improve the security of the website. Some of these will include Input validation, password hashing and limiting invalid logins. Web app security is important as in this day in age, across the internet there is a stupendous number of users who have malicious intent when browsing the web. If a website doesn't have a solid web app security structure, there at risk of being targeted by these users. This leads to numerous problems, one of which being users in general won't trust a site that is easily hacked, leading to site traffic being low.

Examination of the app

User Access

To start off with, the user has excess accessibility to the website, and can see certain fragments of the website that they shouldn't be able to. This can cause issues as certain users may attempt to perform malicious attacks on the site. It's important that this doesn't happen as it is a lottery site which stores a lot of user's data, including the winners. The site may also have access to the prize money which malicious black hat hackers may try to gain control of. One example of this is that users have access to the admin page. This is an issue as anyone can see the current winning draw, they can also create a winning draw allowing them to fix the game in their favour. Anyone who is on the site can also see a list of all the current users on the admin page, this contains their ID, email, full name, and phone number. This is a breach of confidentiality which could lead to the site being sued for a lot of money.

The Login & Register Pages

One massive issue with the site that I noticed straight away was that the login page doesn't work at all. This is a massive issue as users won't be able to check if they have won, in fact they wouldn't even have access to their account after registering. So, if they did win, they would not be able to access the money. There are also major issues with the register page, there is 0 validation for any of the fields, this means that every field accepts any input. This is an issue as users can register accounts with invalid emails with domains that don't exist. Another issue with the register page allowing any inputs is that passwords that users enter may not be secure, for example a user could set their password to "password". This leaves accounts with weak passwords vulnerable to malicious attacks by hackers. If a hacker was able to gain control of a lottery winning account, they may be able to claim the money. This is very easy for hackers to do in this day in age if there is no password validation as they can just use something as simple as the brute force attack to gain access to accounts. Another issue with there being no validation on the register page is that the database may end up with a lot of empty records, or a lot of records containing empty fields. This is unorganized

and can lead to the users account security being weak as it could be missing fields such as their email or phone number.

No encryption or Decryption for the lottery numbers

No encryption or decryption for the lottery numbers is a massive issue for the website as if a malicious hacker was able to hack into the database, they would gain access to the upcoming lottery numbers, allowing them to fix the results leading to them winning in the future. To avoid this, we can encrypt the data, which would drastically improve the security of the database and the website overall.

Security Programming

Implementing Input Validation

One of the first things I made sure to do was to implement input validation into the lottery web app as it had not already been done. Input validation is important when developing a web application as it is essentially testing the input supplied by the user, making sure it is in the correct format, correct length, etc. input validation stops improperly formed data entering the information system, this means that all data that is entered into the website and stored will be of the correct format, keeping things such as the database clean. It is also tricky to detect a malicious user who is attempting to attack software, so if the application validates all user input and makes sure it is of the correct format and not some form of malicious code for example, it will drastically help improve the security of the web app.

Password Hashing

Password hashing was another security mechanism that I implemented that I thought helped improve the security of the website drastically. Most users tend to reuse passwords all the time across different applications, this means that if a different application that the user has an account with ends up getting hacked, it could be possible that the user's account/password on the lottery web application could end up being compromised without password hashing. Password hashing essentially turns the password into another string, that is a lot more complicated and would be harder for an attacker to obtain. The unique randomly generated string is sometimes called a salt, A salt is unique to each user making cracking large numbers of hashes significantly harder, as the time required grows in direct proportion to the number of hashes.

Limiting invalid logins

This security mechanism is something that also helped with the security of the website quite a lot, it is self-explanatory; it essentially involves limiting the amount of invalid login attempts that a user can make before locking the user attempting to log in from entering their email or password. This means that malicious users can't use the brute force attack and keep attempting different passwords until it eventually found one that matched a user. Brute force attacks cause a few issues, it puts users accounts at risk and floods sites with unnecessary traffic. Limiting logins is a popular way to block out the brute force attack.

Role based access control

The last security mechanism that I am going to mention is role-based access control. This was super important for the web application as before I implemented it, anyone could see the admin page which contained details such as viewing the winning draw, creating the winning draw, view all the users, see the logs, and run the lottery. Sorting out role-based access control meant that anonymous users could only see the home, login and register page; regular users can see what is relevant to them such as the account, profile, and lottery page, and only admin can see the admin page.

Evaluation

Security mechanism that could have been implemented

One security mechanism that could have been implemented was a CAPTCHA on the login and register form to make sure that the user was human. CAPTCHA is a set of tools that you can use that can differentiate between users and automated users. The great thing about CAPTCHA is that google has set-up a program (reCAPTCHA), this is free to integrate into web applications. This has led to it becoming the internet's standard CATCHPA programme as it is easy to set up and free. Whilst this does stop automated malicious bots from gaining access to certain parts of the web app, CAPTCHA does also come with its drawbacks. It can be extremely frustrating for users while entering a CAPTCHA if the user thinks they are entering the right answer, when they aren't. In addition, it may end up being difficult to understand for some audiences, in today's day in age a wide variety of people use the internet on a day-to-day basis, with everyone having a different level of knowledge on how to browse/operate the web in a safe manner. Lastly with advances in technology, there is now a range of automated systems, including API's and extensions that allow an attacker to easily get through CAPTCHA's.

Conclusion

Overall, I found the coursework equitably challenging, often I would run into an error and find myself stuck for a few hours tweaking different parts of the code to get the programme to work. One thing that took me a few hours to work out was to do with access management and changing the placeholder names. It took me a while to realise that you had to import and use current_user. I found out that a lot of my issues where to do with me missing certain bits of code from previous challenges, for example I did not realise I was missing a piece of code for the login manager in handling user logins. Later, down the list of tasks when I was attempting access management and logging, I was running into the same error to do with 'anonymoususermixin'. It took me an hour or two to work out what I had done wrong which was just missing out 3-4 lines of code which may have seemed irritating at the time but taught me to always take my time and make sure that each task is completed.