

Secure Sports Betting

MSCS 630

Anthony Connolly

April 24 2022

Abstract: In implementing a secure sports betting app we must allow users to create an account/login and view/place bets. Doing so will require hashing of a stored password and block encryption for the betting data. These will be implemented using SHA 1 and AES respectively. The data will ultimately be stored in a local database. For the milestone, the user interface is fully implemented in react native and the goals and scope of the final project is have been set.

Introduction: Sports betting using mobile applications or websites provides new opportunities for prospective sport bettors but also new risks. Undesired outside access to one's account may provide unwarranted access to a user's payment details, and allow unwanted stakes to be placed with the user's name. Additionally, a person may not wish access to previous bets, so those must also be made private. We will explore how existing mobile betting applications keep a user's account and profile secure so that we may begin to build a similar, secure application that emulates sports betting. We will also explore how we can encrypt and decrypt a users betting data and store it locally.

For an implementation of this project as a consumer product, I would avoid storing data locally and use something like Firebase, which has it's own encryption system that hashes and salts passwords. For this reason, I will be storing account and bet data locally and exploring effective ways to encrypt both types of data.

Background:

Two types of data must be encrypted/hashed for this application, the password for storage in the database and the betting data. Typically, passwords stored in a local database should be hashed and salted. Hashing and salting passwords correctly prevent two users from having the same hashed password stored in a database, thus also helping to prevent hash-table attacks. (Arias) Some examples of hashing algorithms used for hashing and salting passwords are SHA 1, SHA 2, bcrypt, etc.... For my application it would be sufficient to implement SHA 1 hashing and to salt the password for storage in the database.

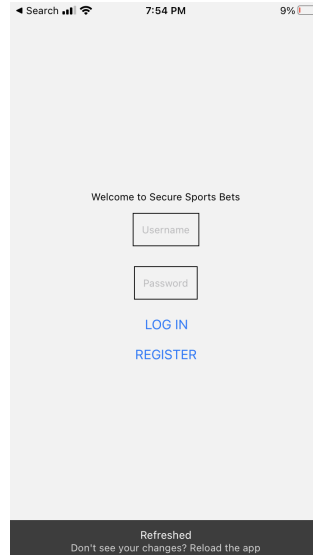
For betting data we must encrypt plaintext since all the necessary data for the bet can be converted to and from plaintext. Some example of such algorithms are AES, Blowfish, and RSA. (Simplilearn) AES at 128 bits will be used for encrypting our betting data.

Experiments:

Another approach taken to help chart a course for our secure application was experimentation with existing mobile applications, mainly Draftkings and Caesar Sportsbook, to see what measures exist to protect a user and how the generally apps function.

Through experimentation and research with existing sports betting apps, it seems to be a standard of the sports betting application market to log a user out after a set period of time from the last login. This would require the user to log back in to access funds or place a bet. This measure would help mitigate some issues from someone's device being compromised, such as placing unwanted bets or draining the account owner's funds. Another measure is a limit on deposit amounts. Surprisingly, two-factor authentication is not required on either tested application.

Methodology: For the first milestone, the objective was to make a functional GUI for our secure sports betting application. This GUI has three essential components, which will be further discussed in detail, a sign up screen, a log in screen, a betting screen, and a previous bets screen. This GUI is created using React-Native and we will use SQLite in the future to store the data locally.



Search 7:54 PM 9%

Welcome to Secure Sports Bets

Username

Password

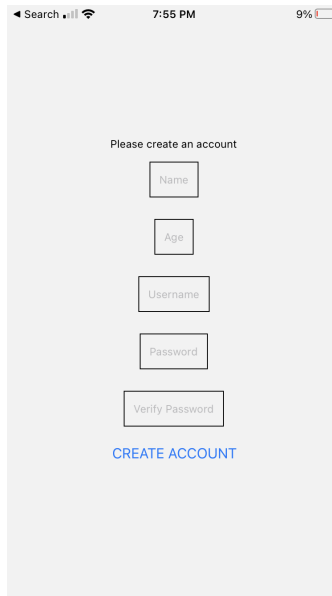
LOG IN

REGISTER

Refreshed
Don't see your changes? Reload the app

This is a mobile app login screen. At the top, there is a status bar with a search icon, signal strength, time (7:54 PM), and battery level (9%). The main heading is "Welcome to Secure Sports Bets". Below this, there are two text input fields labeled "Username" and "Password". Under the password field, there are two links: "LOG IN" and "REGISTER". At the bottom, there is a dark grey bar with white text that says "Refreshed" and "Don't see your changes? Reload the app".

The signup screen UI is quite standard. We will require user's to input their first and last names, age, username, and password. To take in such data, we use a text box for the fields of first name, last name, email address, password and age. We also require user verify their passwords. There is finally a "Create Account" button which will ultimately both trigger SHA1 hashing and salting of the user's data and store it in the database. At the moment, if the user has entered valid data for all of the fields, we will simply be taken to the betting screen.



Search 7:55 PM 9%

Please create an account

Name

Age

Username

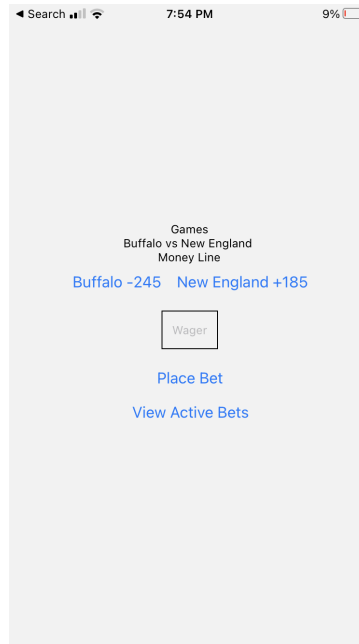
Password

Verify Password

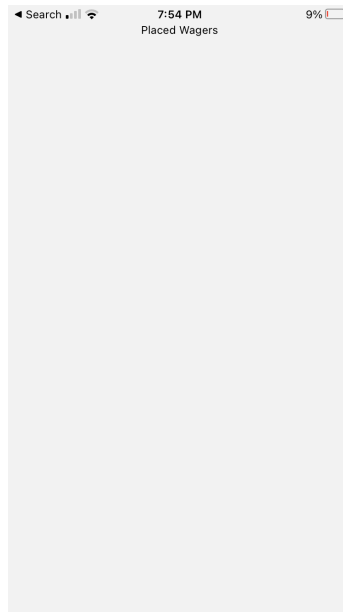
CREATE ACCOUNT

This is a mobile app account creation screen. At the top, there is a status bar with a search icon, signal strength, time (7:55 PM), and battery level (9%). The main heading is "Please create an account". Below this, there are five text input fields labeled "Name", "Age", "Username", "Password", and "Verify Password". At the bottom, there is a link that says "CREATE ACCOUNT".

The Log In screen prompts the user for their email address and password with fields. The submit button currently takes the user to the betting screen upon successful filling the fields with valid data. The user may also press the text link to the account creation page.



The betting screen presents the user with games to bet on. The UI presents the user with a text field for each game, to place a bet amount. The user is also presented with two buttons for each end of the bet's moneyline odds. To place the bet, the user will input an amount, press the button for their bet and press the submit button at the bottom of the screen. Doing so creates an alert UI saying the bet was placed successfully. In the future, this will convert the bet details to plaintext and use AES and place it in our local database.



Upon full implementation of our bet encryption and app, the loading placed bets screen will decrypt pull our bet data from the database, decrypt the details, and display the details for each bet.

Conclusion: At this point in the process of building the application, enough research has been done to have a clear flow of work to complete the application. We also have an extremely solid foundation of a fully completed user interface. The next steps are implementing encryption and decryption using SHA - 1 for password storage and AES 128 for betting data. We also must build simple locally stored databases for both using SQLite.

References:

Arias, Dan. "Adding Salt to Hashing: A Better Way to Store Passwords." *Auth0 - Blog*, 25 Feb. 2021, auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords.

Simplilearn. "What Is Data Encryption: Algorithms, Methods and Techniques [2022 Edition]: Simplilearn." *Simplilearn.com*, Simplilearn, 4 Mar. 2022, <https://www.simplilearn.com/data-encryption-methods-article>.