David Smith, Cole Morrison, Anthony Greene, Dayne

Professor Singletary

CEN3024C

11/6/2024

FinancialClarity Security Documentation

For FinancialClarity we utilized CORs to help secure our application. Which was used to define a set of domains, schemes, or ports for application's resources. How CORS works is in the browser it will send a preflight request that will determine if the actual request is safe. This makes it so that applications that want to see our information are safe and can also get past the same origin policy. This was done because certain frontend applications can't have access to the project's APIs through backend application. It is very important that the application's APIs are secure from unknown applications that can access through the backend. We also added authentication for the user by salting and hashing the user's password. Salting will make sure that even if multiple users have the same password each password's hash will be unique. Also making it more difficult for attacks because they would have to re-compute the passwords using the salt of each user.

After the launch of this project in the future we would like to apply the principle of failing securely. We understand that no matter how much security we have for the project that errors and attacks will happen to the application. So, when they do happen, the system will go to a default secure state instead of fully exposing vulnerabilities. What does that mean if with help make sure sensitive is leaked by make sure the application system doesn't crash. But this will make sure that the data isn't leaked or exposed when it crashes. Hashing and salt will help with this by making sure even if they do get through, they must decipher the hash to get the user's

information. We also took a measure in this by stopping the application from running entirely in case of this or any error issues. Speaking of vulnerabilities, we would also want to minimize our attack surface from threat actors. We have already taken a step into doing that with the project by only using necessary code. Such as getting rid of the duplicate APIs and deleting lines of unnecessary code to our application. Doing this prevents any errors and opening vulnerabilities to attack. For our services, DTOs, and repository we would like to do this for our API endpoints, so no sensitive data is leaked. Lastly, we would want to have a separation of duties from the user. Such as least privilege for the user so they only those who are permitted are allowed to see all data. This will also help create groups who can access that data and those who can't. This will further minimize misuse of the system, access to all data and deletion of data. So there for only the user can see only their data and nothing else.

As far as our approach to implementing the security for the project we went with a security as a code approach. For we used a lot of predefined code in spring boot to help with our security for the application. We also used some automation remediation for our code with GitHub for the use of testing of errors and vulnerabilities. Also utilizing maven in spring to hold and update any dependences we needed. While using booth spring and GitHub to find any real time errors within testing or code. GitHub was very helpful in testing code whenever a commit was pushed with the use of overflow files. So, the code would immediately test after any changes were pushed and give a reason why it didn't succeed.