Threat Modeling Assignment

Partner & Interviewee: Arni Tiwari

1: **The Interview**

1) **What is your application designed to do?**
   The application is designed to let a user create an account and see the list of users who have made an account..

2) **What groups of users are you expecting?**
   I'm expecting general users with limited access to my application. Other unauthenticated users wont have access to internal pages. I am the only person who has admin access.

3) **How does a user register? Login?**
   A user can register by clicking on the Registration link at the top of the application. This takes them to the Registration page where they have to enter a username and password (8+ characters containing a capital letter, a lowercase letter, a number and a special symbol). Once registered the user is taken to the login page where they can enter their credentials. For just logging in, the user can go to the Home page and enter their credentials. The first page that loads on starting the app is the login page.

4) **How does a user delete their account?**
   The user would have to go to MongoDB Compass and delete it manually

5) **How is data stored?**
   Data is stored in mongodb. Passwords are hashed and salted and stored in mongodb.

6) **What documentation is there?**
   There is package.json and package-lock.json for the frontend and backend which contains the list of dependencies and other libraries used. There is also the SBOM cyclonedx documentation and dependabot vulnerabilities.

7) **What features are currently partially complete?**
   In the branch you are viewing the oauth is not set up.

8) **What issues and failures have you currently identified?**
   Someone can brute force the application because there are no checks to see the number of times an incorrect password is being entered. They can do credential stuffing. There could be insider threats and knowledge that can help break the application. Bypassing access controls can lead to integrity damage on the data and can give access to sensitive user info like passwords.

9) **How have you taken care of the backend security of your app?**
   I have encrypted passwords and used https to encrypt the paths going and coming to the backend. This is what I have taken care of and I think there are a lot more areas I can protect if given more time and resources.

10) **How have you handled frontend security of your app ?**
I have used https to encrypt the paths going and coming to the frontend. In addition I have implemented cookies to be sent to authorize users and protected routes to prevent unauthorized users from accessing any page. This is what I have taken care of and I think there are a lot more areas I can protect if given more time and resources.

11) **What more features are you looking to implement?**
I was going to implement an oauth for my app.

12) **What technology in your stack do you think you want to change. And why?**
If the application requirements involve complex querying or transactions that might benefit from a more traditional relational database, switching to a relational database like MySQL would be better.

13) **Is your data encrypted in the storage?**
Yes

1.B: **Initial Manual Analysis**

1) **What files exist? What do they do?**
The root folder of the application has a folder for frontend and one for backend. And there's a .gitignore file that has info on which files to not include on github.

2) **Is there any sensitive data stored on the github?**
No the github doesn’t have any sensitive data

3) **What technology stack is used?**
Technology stack used is JavaScript, HTML & CSS as given on github. The Code uses Node.js framework for backend and React.js for frontend. The database is at MongoDb Atlas.

4) **What analysis tools are being run? What do these reports say?**
I don’t see any active analysis tools on github. However I put snyk scanner on the repository I forked and they found the following major risks: Regular Expression Denial of Service (ReDoS)

5) **What documentation is available?**
Other than the package.json files in both frontend and backend, the github codebase has SBOM files for both parts of the Full Stack Webapp.

6) **What database technology are they using? How are they defining their data structures?**
They are using MongoDb for their database. They don’t have any extra ORM for their data models. They used mongoose models. I found no other data structure getting used other than arrays.

7) **What pages are available in the frontend? What pages are available in the backend?**
Pages available in FE:
   - /home
   - /landing
   - /registration

   Pages available in BE:
   - /account/register
   - /api/createNewUser
   - /account/login
   - /account/users
   - /account/users/:username
   - /api/contacts

1.C: Initial Manual Dynamic Analysis

1) **What can a non-authenticated user do?**
A non authenticated user can go to the login page http://localhost:3000/home and the registration page http://localhost:3000/register

   a) **List available pages and routes on the frontend. Test them all and record your results**.
   A non authenticated user can go to the login page http://localhost:3000/home and the registration page http://localhost:3000/register

   b) **List available routes on the backend. Test them and record your results**.
   A non authenticated user can visit the register route http://localhost:8000/register and list of users route http://localhost:8000/user

2) **How do you create an account? How do you login?**
You can create an account by going to the registration page and typing a username that isn't taken and a password that is of minimum 8 characters and has at least one capital letter, one lowercase letter, one number and one special character.

   a) **As a logged in user, test all available pages and routes. What did you get access to?**
   We get access to the landing page which consists of the list of all usernames of users in the database

   b) **Similarly, test the backend.**
   We can access the login route, register route, create a new account route, display all contacts route, and get all users route

3) **Where can a user inject data? Remember to consider frontend, backend, and error routes.**

A user can inject data in error routes and also via input forms or query params on request.

4) **Where is stored data shown to users? How?**
Stored data can be seen only if one has access to the database or through connecting to the database and using the api routes in postman.

5) **How is the database secured?**
Data-at-rest encryption is done for passwords, https is used for communication between MongoDB and the backend and a jwt token is required to access individual user data.

Other Analysis

There were deleted certs and .env files in older commits which pose a security risk. It is a fact that whatever code we push on the internet stays on the internet. When I had first scanned the project with Git Dependabot, it did not catch this bug. I found this during manual analysis which is a proven fact that both static and dynamic analysis of the code is important.

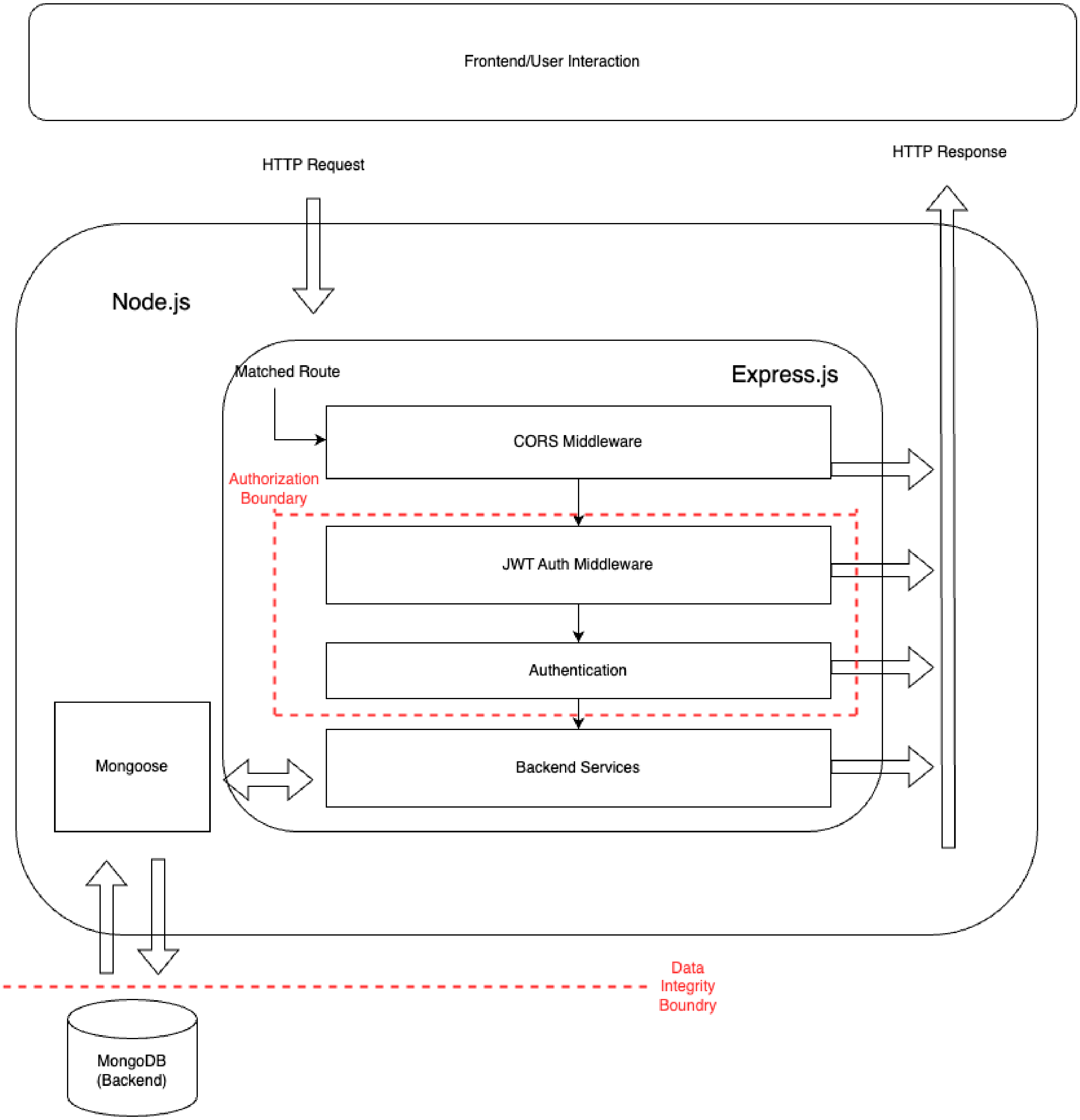
The list of users' code is visible to all and everyone regardless if they have authorization or not. It is important to keep this info hidden from non-registered users.

While this use case doesn’t require solid file management, it could get cumbersome to manage the code base if more features are added. Most of the backend’s routing and api’s are mentioned in the base file itself.

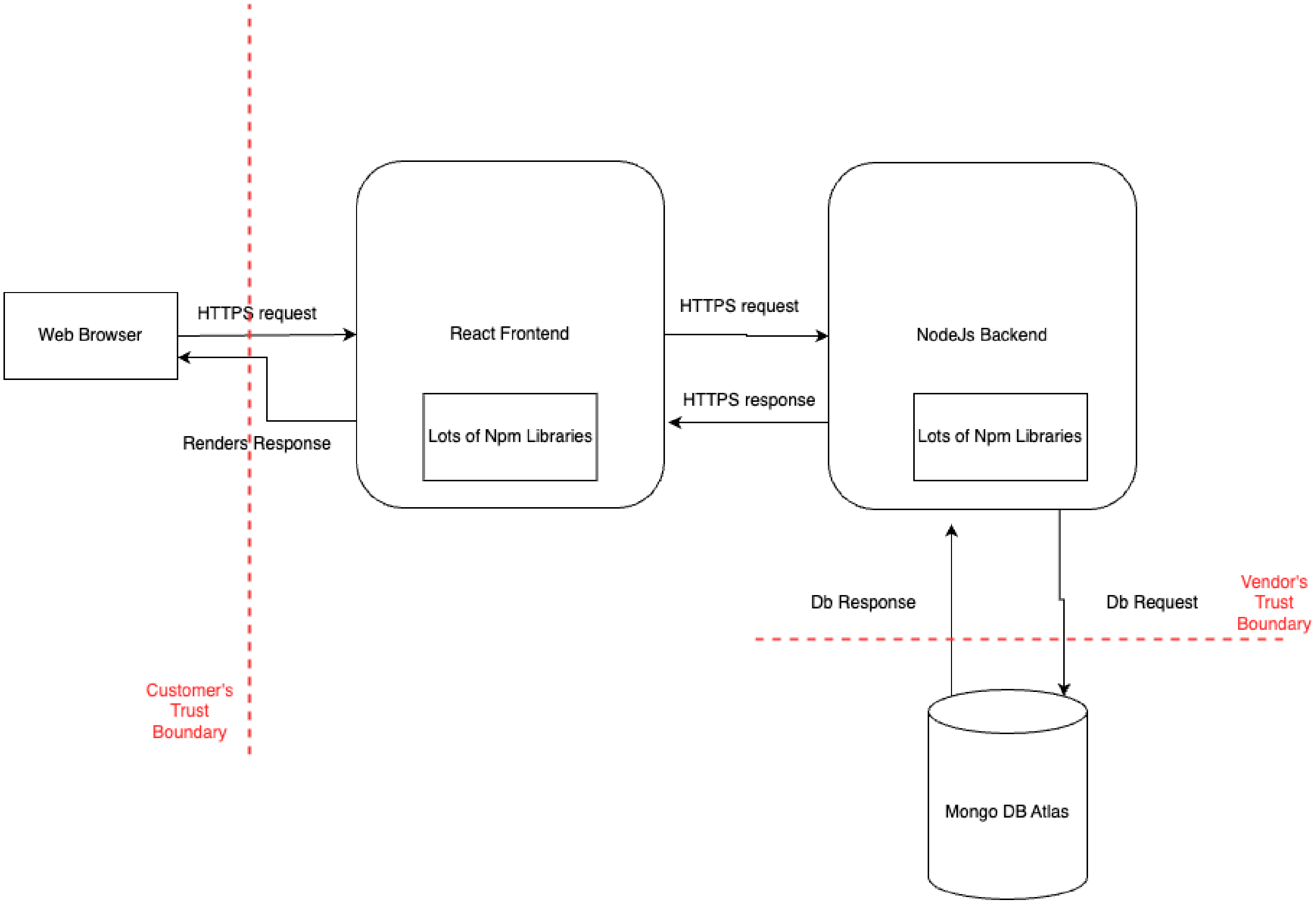
A Synk or other vulnerability scanner tool integration to the project would also be beneficial in the long run.

2 & 3: Application Diagrams w STRIDE element

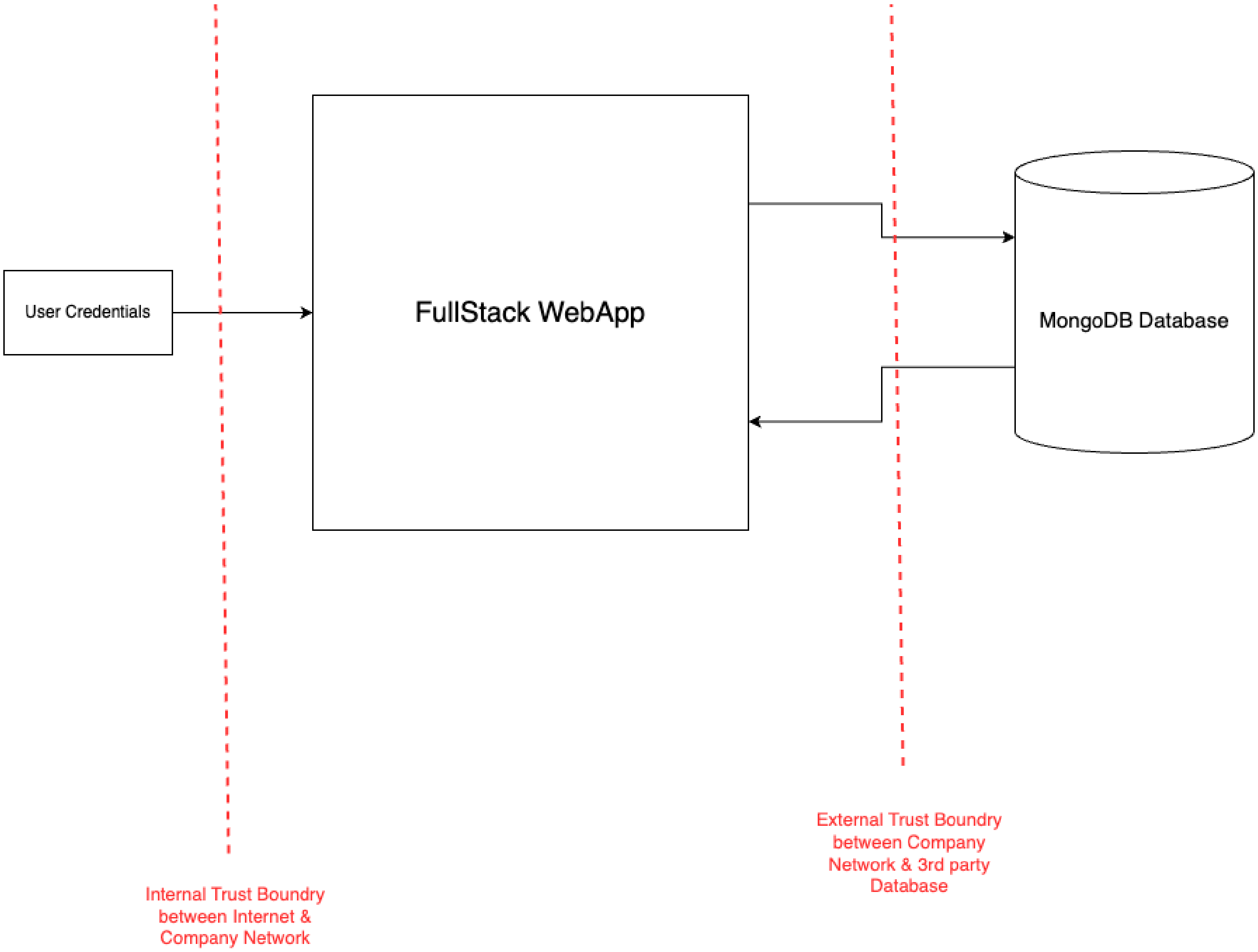
2.A: Application Diagram



2.B: Dependency Diagram



2.C: Data Flow Diagram


User Credentials
FullStack WebApp
MongoDB Database
Internal Trust Boundry between Internet & Company Network
External Trust Boundry between Company Network & 3rd party Database

4: Security Cards

Identify Direct and Indirect Stakeholders

1) **Identify direct stakeholders in the system. Groups should also identify indirect stakeholders.**
   The system has the following direct stakeholders: Users, Admins(Developers)

   The system has the following indirect stakeholders: The database (MongoDb) integration is an indirect stakeholder since we are using it's services.

2) **Identify Human Assets at Stake**.
   According the Human Impact Cards, the following are at stake in this application:

- Personal Data of users
- Financial Wellbeing, if DDosed we could get charged for using MongoDb’s free services.
- Emotional wellbeing of users

3.a Identify potential threats to the system

Technological attack, manipulation or coercion attack, indirect attack & multiphase attack.

3.b Consider a series of threats by (randomly or purposefully) selecting sets of cards; these sets should contain cards from at least two different dimensions.

- Curiosity or Boredom Adversary Motivation Card, Technological Adversary Method Card.
- Money Adversary Motivation Card, Multiphase Adversary Method Card.
- Expertise Adversary Resources Card, Attack Cover up Adversary Method Card
- Desire or Obsession Adversary Motivation Card, Inside Capabilities Adversary Resources Card, Manipulation or Coercion Adversary Method Card

3.c Record and discuss which 3 threats to the system are the most relevant to the system.

- DDos Attack using the regular expression exploitation. The backend code uses regex expression to validate and Inefficient Regular Expression Complexity can lead to ddos attacks trying to guess password or username combinations.
- The system is Secrets Leakage Attack vulnerable due to past git commits exposing old keys and certs. An attacker could misuse credentials to host a phishing attack using our keys to misdirect the users. They can also get into our mongodb instance and steal data.
- Information Disclosure:Depending on how errors are handled and reported, an attacker might be able to gain insights into the workings of the underlying system

5: **Executive Summary**

There are a number of security firms out there in the market but not every firm would be as strategic as we are. Our team comprises industry veterans with a proven track record in application security, risk management, and compliance. This ensures that your application is not just secure but also aligns with industry standards and regulations. We've honed a meticulous three-step check process to guarantee your application's security. Initially, we fortify the environment in which employees work. It’s important to keep our data secure so that no other party can exploit the work we’re doing. Second, we align your application with IEEE's secure protocols, identifying and rectifying potential vulnerabilities. These enhancements are then scrutinized by our White Box testers to make sure there aren’t any loopholes left in our evaluation.

Recognizing that each application has unique security requirements, we offer customized solutions that address your specific challenges. This includes comprehensive vulnerability assessments, penetration testing, and security architecture design. Each issue is approached with a fresh perspective, always with stringent security protocols in mind. We see our role not just as service providers but as strategic partners, committed to empowering your team with the knowledge and tools necessary for sustaining and advancing your application's security, thereby cultivating a culture of security within your organization.

Our team makes use of state-of-the-art technologies like Snyk and Nessus, we employ multiple tools simultaneously to optimize bug detection and remediation processes before your app goes to deployment. Any identified bugs are securely communicated to your development team, either in person or via secure, encrypted file transfer protocols upon request.

We take the security of your data seriously. Recognizing that a firm of your stature may frequently attract attackers, we implement multiple layers of defense to make sure the attacks are stopped at the initial attack vectors. All and every data is segmented, building the application’s defense even further. Our staff is rigorously trained in security practices, including the safeguarding of MFA credentials. Data encryption is a standard practice, and the keys will only be available to your admins where the keys are refreshed annually.

The logs of the application are constantly monitored and the application traffic is regularly analyzed. Any suspicious activity will be reported to your security team. We also offer protection against DDoS attacks, ensuring uninterrupted service for your customers.

We believe that security is not a one-dimensional endeavor. Our services are designed to guide you through the intricate web of data protection laws, guaranteeing compliance across the board. We focus on all and every aspect when it comes to securing your organization. From professional 1:1 risk analysis training to online security training courses, we provide all the services your employees need to gain expertise in security and risk management.

In choosing our consulting firm, you're not just securing your application; you're investing in a partnership dedicated to the enduring success and resilience of your business. Our commitment to excellence, customized approach, and proactive risk management strategies ensure that your application remains secure, compliant, and ahead of potential threats. Let us help you transform your application security posture from a point of vulnerability to a competitive advantage.