Software Security (CSE 545) MCS Portfolio Report

Chirag Bhansali
Arizona State University
Tempe, US
cbhansal@asu.edu

I. Introduction

This is a portfolio report for Software Security (CSE 545) course taken in Spring 2019 semester. For this course, we had to create a prototype banking application, using Java, Springboot and Maven along with My SQL. The application has features like Debit and Credit amount,transfer money to other users and a few critical security features like Authentication, Authorization and protection against common hacking attacks like Broken Authentication, Sensitive data leakage, weak passwords.

The banking application we created has two types of users : individuals and merchants. Both of them can do debit, credit and transfer money to other users or merchant using either the receiver's phone number or email id, which is registered with the banking application. They can also view their transaction log. Individual users can create as many as 3 accounts either checking or savings accounts and merchant can create 2 accounts one checking and one savings accounts for transactions.

On the banking side, there are 3 types of employees: Tier1, Tier2 and Tier3 or Admin. The application can have only one admin user, who can create, modify or delete employee's account. The admin can also authorize or decline employee requests for change in information and lastly, can view system logs which contains information about employee login, logouts, customer transactions and insert and deletion operations by the banking employees.

Tier1 and Tier2 employees can create, approve and decline transactions. Tier1 can approve transactions upto 1000 dollars and Tier2 any amount above 1000 dollars. Tier1 can also issue cashiers cheque. Both of them can view, modify, create and close customer accounts and information about the customer accounts.

The customers of the application can also schedule an appointment with an employee in the bank for which they are required to enter an OTP (One Time Password). The application uses a self-signed certificate and SSL/TLS connection for communication purpose between the client and server, thus all the information is in the encrypted format during communication. In the database, also all the personal information like email id, first name, last name,phone number, SSN are stored in encrypted format using AES256 function present in MySQL.

The application was deployed in the AWS (Amazon) EC2 server, with the link to the website available to follow classmates, professor and the TA, with an assigned bunch of students trying to break and hack into the application and find any possible vulnerabilities or weakness which could be exploited or misused to do unsolicited activities.

II. Explanation of the solution

A. Security Designs

1) PKI and Digital Certificate: We used PKI (Public Key Infrastructure) and Digital Certificate for communication between the client and server. The certificate we used in the application is a self-signed one, which helps in keeping the data secure, by encrypting the data and communication, so a hacker can't gather any useful information by just capturing the packets.

2) OTP and Virtual Keyboard: For scheduling an appointment with a banking employee and transferring money to other users or within different accounts of the customer, an OTP is sent to both the phone number and the email id provided by the user, with the customer getting three attempts to enter the correct OTP. The virtual keyboard is used for preventing attacks like Keystroke logger.

3) Login Controls: Each and every user and employee is given a unique user id, with conditions set on the length and complexity of the password that a user can have, preventing against weak passwords like "password" or "123456789" to be used by anyone for login.

4) XSS and CSRF: The login page has a unique token which refreshes after a set amount of time. In CSRF (Cross Site Request Forgery), a hacker sends a page which resembles like the original web page except the enter/submit sends the information to the hacker, which can be misused. The unique token prevents that attack since the system validates the token and if the token isnt the same, the user is logged out of the system.

For the XSS (Cross Site Scripting) attack, any malicious user or hacker would try to insert/ enter scripts possibly JavaScript code in the textbox, which would execute at the server side and expose the side for the user to hack into. We prevented such sort of a scenario, by having validation at both the client side as well as the server side to ensure that the data entered by the user is genuine and in proper format. We used regular expressions for every sort of data, like only numbers for phone numbers, names can be only character and no special characters or symbols or numbers can be entered in those fields. A proper format for emails and SSN.

5) Data Masking, Encryption: As mentioned above, we used SSL/TSL for communication and digital certificate for encrypting the data/records between the clients and server, making it difficult to decrypt the data and gather information just by swooping the packets.

For storing the data, we used a RDBMS called My SQL, where all the personal information of the user like first name, last name, SSN, email id, phone number are stored in encrypted format. We used AES256 for this purpose.

6) Logging and SignIn History: One of the major vulnerabilites that is quite exploited, especially by internal employees is having no logging system for the application, which would allow them to make changes in the application without anyone ever knowing who or how the action was done. It has featured in the OWASP Top10 for quite some time. We took steps against this type of attack by having a logging system, which would track the login, logout of the employees, all the approval and decline operations as well as the transactions done by them especially create and delete operation. Plus, the logging part is only visible to the Admin user who cant delete or modify the logs, thus prevent any chances of logs being modified or deleted even if the Admin user account is compromised.

Every page validates and verifies the operation done by the logged in user, thus preventing tier1 or tier2 users to do any unauthorized actions like Tier1 wouldnt be able to approve or view transactions accounting to more than 1000 dollars as well as the Tier2 cant view the ones which are less than 1000. Tier3 or the Admin wouldnt be able to view any transactions or approve them or view customer information.

B. Banking Functions

Just like any US banking application, this banking application had to also provide all those services and features like Debit, Credit funds, transfer funds either within the different accounts of the same user or to other user using the email or the phone number of the receiver.

1) Technical Help and Support: The users of the application i.e. the individuals and the merchant have an option to book/ schedule an appointment with the bank employees with proper timings and slots of their convenience.

Since, this is a prototype application, with the end users generally being the fellow classmates and TA. Mine and Deputy leader's email id was provided for any sort of support with regards to application not working, or any technical help or feature not working as per the expectations.

2) Accounts, Statements: The users can create a finite number of accounts either a checking account which can be used for both debit and credit funds or a savings account which can do transactions only to the respective user's checking account only. Users can also download statement either for the entire month or a specific date duration they choose.

The banking employees can also create, modify and delete accounts based on their access level. Tier1 can only view customer information. Tier2 can view, modify and create accounts.

3) Money Transfer: The banking customers can either transfer money to other users via email, phone number or they can transfer the money to any of their other accounts, subject to approval from the tier banking employees.

Just as a precaution, the users cant transfer more than the amount in their accounts. This helps prevent the amount in the account to go in negative.

4) New Users and Employees: For setting up a new account in the bank for the very first time. There are two choices, either the tier2 employee can create the new account and userid and password for the customer or the customer can create a new account using his/her details like name, SSN, phone number and email id. To prevent users from creating redundant accounts, we mapped each account with their SSN, email id and phone number. So that each customer can possibly have only one account for use. The Admin has the right to create, modify and delete banking tier employees.

III. Description of the result

This project was a prototype application with end-user being the fellow classmates, professor and TA. The application was deployed on the Amazon's cloud services, which helped in making the access to the website and application available for 24*7, with multiple users using the applications simultaneously.

The overall project and application didnt had any major shortcomings and flaws which could be exploited by the assigned group of students who tried various methods and techniques to try and hack, and break the application. We took measures to prevent the application against various kinds of attacks right from weak passwords, broken authentication and authorization, XSS, CSRF, sensitive data leakage amongst others.

We also had OTP and virtual keyboard to prevent Keystroke tracking and prevent password or userid from getting stolen as well as to prevent unauthorized transactions.

Verification and validation of the data entered as well as the user who is entering the data helped in preventing unauthorized access and usage of the account or information. Encrypting the data helped in preventing exposure of sensitive and private data of both the employees and the users. Logging and action tracking helped in zero down the exact time and user who did that action and prevented internal employees to misuse the customer accounts, information and money.

Lastly, this application helped the users, to perform the basic banking tasks like debit, credit money, transfer money, create new accounts, delete accounts, schedule appointments with bank employees. And the banking employees, based on the tier level to approve/ decline transactions, create accounts and user and modify, create and delete bank employees, view logging history.

IV. Description of my contribution

A. Requirement Gathering, Design

I was responsible for gathering the initial requirements and materials for the group project. Understand the problem, communicate with the TA and professor, clarify any ambiguous problem statements and requirements. Did the documentation work for the project, created the SRS (Software Requirement Specification), ER diagram, Use case and Gantt Chart for the entire project'schedule/timeline.

Created the GitLab repository where me and my fellow teammates pulled and pushed the changes for the application and worked in a collaborative manner. I was responsible for the integration of the modules and did the critical unit and integration testing for the built ones, before and after integrating the same in the main application.

B. Weekly Reports and Meetings

We followed the Agile Model for developing this application, with Sprints of 2 weeks. We had meetings after every class, where I communicated and resolved any doubts or queries of my fellow teammates, created and sent meeting minutes to the TA and professor, which had info about the meeting objectives, discussion topics and the teammates who attended the meetings, along with the current and future progress of the project.

C. Modules

1) Employee Dashboard: After an employee has successfully logged in into the system, they are presented with a dashboard, where tabs for various kinds of operations depending upon their level of authorization, i.e. whether they are Tier1, Tier2 or an Admin. The tabs redirected the authorized user to pages, where they can approve/decline transactions both critical and non-critical transactions, approve/decline account creation request, create a cashiers cheque, create, modify or delete an account, create, modify or delete an employee as well view customer information.

2) Schedule Appointment: Both the individual and the merchant customers can schedule an appointment with a banking employee, where they can choose the reason and the date of the appointment. The system also sends an OTP to the user's email and phone number, without which they can't book/schedule an appointment.

3) Search Employee and Customer: An admin can view, modify, search and delete an employee. Created the page and the interface for the same. Tier1 and Tier2 employees of the bank can view the customer information based on the customer's account number.

D. Server and client-side validation

For validation on the client-side, used jQuery for verifying and validating the inputs and ensuring that no scirpts or improper text, symbols are entered by the users. The application user is prompted with an appropriate message and asked to correct the same. Added these validations in the backend as well, before sending the information to the database, with the help of regular expressions.

E. Deployment, Maintenance and Support

I deployed the application after through testing and requirement validation and verification on the AWS(Amazon) EC2, and sent the website link to the assigned classmates who would be testing the application and try to find flaws, shortcomings and vulnerabilities in the application.

Provided the maintenance and support assistance after the deployment and helped others, if any issues regarding application not accessible, functionality not working properly were faced by anyone.

F. Final Presentation and Report

Created and presented the presentation which had a brief description of the project, the flaws and how to resolve them were there. The final report, contained the entire task list, the vulnerability report and the final design document of the project, the ER and the Use Case diagrams.

V. New skills acquired

This project was very helpful, where I learnt not just technical skills and attributes but being the group leader of the project learnt a lot of management and leadership skills and techniques as well.

1) OWASP Top 10 Vulnerabilities : The Open Web Application Security Project, learnt about the most common vulnerabilities that majority of the web application deployed and used today have and ways and methods via which those vulnerabilities are exploited and misused by malicious users.
2) Banking Functionality: This project taught me the basics and the working behind common banking activities, i.e. the debit and credit amount, the different types of the accounts which are there, how they are different and the purpose they hold. Learnt about the inner workings of Money transfer, ways money transfer can happen without the need to visit the bank, and how email id and phone number can be used for the transfer. Also, learnt about the fields and the basic information required for opening a new account.
3) Management Skills: Managing the group of 10 members with varying mindset and thinking, learnt about the essential leadership skills, managing them and ensuring that the tasks are done within the set time frame and how to be good leader by helping other fellow teammates when they are struck with some problem.
4) Technical Skills: This project taught me a lot of newer tools and technologies like Java, SpringBoot, Maven, AWS EC2, Unit and Integration Testing, along with their significance and working. The project also gave me a hands on experience with Agile SDLC and why its better than other SDLC techniques when the project requirements are dynamic and sprints are short and fast.
5) Security Skills: Learnt about the encryption, decryption of the sensitive data, its importance and techniques to ensure that the sensitive data isnt lost or leaked. Learnt about SSL/TLS and PKI, why and how they are used to

ensure that the communication is secured and the data is in encrypted format. A few basic hacking techniques and methods to try and exploit common vulnerabilities in a web application especially using the API calls to try and use brute force method and gather information about the customer and the employees.

VI. Team Members

The project had 10 members in it. Other members are Pranay Jagtap, Srajan Gupta, Lei Zhang, Alex Nou, Ramanuj Bhattacharjee, Tanmay Manolkar, Amisha Thakkar, Shachi Shah, Sai Pranav Rangabhatla

References

[1] Owasp.org. OWASP Top Ten. Available at: https://owasp.org/www-project-top-ten/

[2] Garden, H., HowStuffWorks, Money, Finance and ATMs. How Banks Work. HowStuffWorks. Available at: https://money.howstuffworks.com/personal-finance/banking/bank1.htm

[3] Thalesesecurity.com.Thales eSecurity: Cloud and Data Security | Encryption | Key Management | Digital Payment Security Solutions.Available at: https://www.thalesesecurity.com/faq/public-key-infrastructure-pki/what-public-key-infrastructure-pki

[4] Amazon Web Services, Inc. How to Deploy Code to a Virtual Machine – AWS. Available at: https://aws.amazon.com/getting-started/tutorials/deploy-code-vm/

[5] Medium.SDLC Models Explained: Agile, Waterfall, V-Shaped, Iterative, Spiral. Available at: https://medium.com/existek/sdlc-models-explained-agile-waterfall-v-shaped-iterative-spiral-e3f012f390c5

[6] Spring. Available at: https://spring.io/guides/gs/spring-boot/