Developing a Skelton Secure Banking System: Individual Report

Anil Kumar
Arizona State University
akuma254@asu.edu

Abstract— **Software security is a prime component taken into consideration while developing any system. It's a system-wide concern that covers both security mechanisms and design for security. Security Issues are covered in all the stages of the Software Development Life Cycle. In this project, we developed a skeleton of a secure banking system with limited functional, performance, and security requirements for secure banking transactions and user account management and performed various security testing techniques to exploit vulnerabilities in the project.**

Keywords— SSL certification, OTP verification, cross-site scripting, SQL injection, Session Management, Security, Software Development Life Cycle

I. Introduction

As part of this CSE 545: Software Security course's team project, we need to develop a banking system with limited functionality while considering software security requirements at each and every stage of the Software development lifecycle. Multiple users that include both customers and bank employees should be able to securely use the banking system from any place and at any time. And to develop the system, we followed all the standard stages of the Software development life cycle.

A. Planning and Requirement Analysis

It is the most crucial and fundamental phase of the project. The project requirement document was analyzed and all the functional and security requirements for secure banking transactions and user account management were listed out. All the stakeholders and their access levels were quantified. Identification of risks associated with the project and quality assurance requirements were also done in this phase. Technical feasibility study performed to define and finalize the technical approach to be followed to accomplish the project.

B. Defining Requirements

Requirements were provided in the course material. Doubt sessions were conducted to clarify the requirement to all team members. Weekly reporting and meetings were scheduled to synchronize the work between team members.

C. Designing and System Architecture

During this phase Architecture, Programming language, User Interface, and platforms were discussed. The user interface and basic backend prototype were developed at the initial stage. Feedback from all the team members was taken to divide the further tasks among all the members.

D. Development

In this phase development of the project was done by following the requirement documents and using the tools and technologies that were decided in the previous phases. Different team members contributed to the different modules of the project and code was synchronized using a common repository.

E. Testing

Although testing was performed during the development phase, this phase was only focused on testing all functional and security requirements. Various defects were reported, tracked, fixed, and retested until the project met the quality standard defined in the requirement document.

F. Deployment

This was the final phase of the project where we finalized AWS to host the project. A user guide was prepared to help users to navigate and use the banking system.

Resources used in this project:

	Resources
Programming language	Python, HTML, JS, Golang
Integrated Development Environment	VS Code, Python Environment
Python libraries	Flask, logging, JSON, cors, boto3, pyotp
Technologies	Hyperledger Fabric

II. Solution

The users of the banking system are classified in the following four categories according to their roles:

A. *Users*

This category covers all the bank employees and they are further classified into 3 groups:

- Tier-1 employees: responsible for assisting the customer with various banking operations such as initiating fund deposits, issuing cashier cheques and transferring money, etc.
- Tier-2 employee: Responsible for authorization of critical operations. Transactions exceeding $1000 in a day fall in the category of critical transaction. These employees have the authority to create, modify and close customer's accounts.
- Tier-3 employees(Administrators): Responsible to create, maintain, delete and change all the internal/external users.

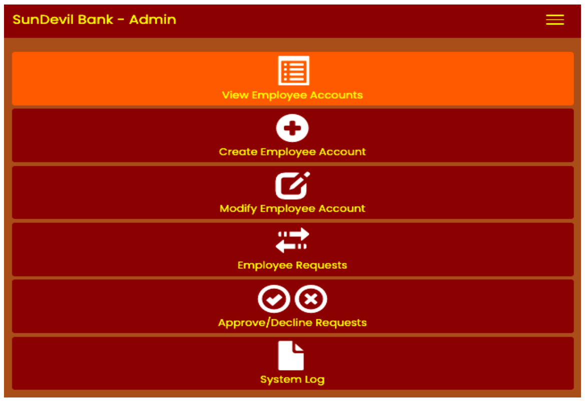

Fig. 1. Admin page

- External Users: Individual Customers, each of them can have checking, saving, and credit card with common functions, like fund transfer, credit, and debit from accounts.

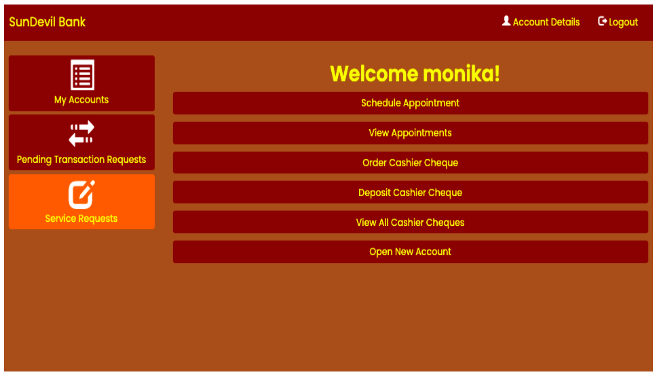

Fig. 2. Customer Home page

B. *Security features:*

Web applications are considered high-priority targets of perpetrators. These features help to safeguard the banking system against different security threats. The following features are implemented to prevent attacks on the banking system.

1) SSL certification: SSL stands for Secure Socket Layer. SSL certificate is installed over the AWS server to provide secure communication. It encrypts the data that's being transmitted between server and browser and provides authentication for the banking system. It prevents hackers from stealing user data.

2) OTP: One-time-password is implemented using the python library *pyotp*. Customers are required to verify OTP while fund transfer, information update, and issuing cashier cheques.

3) Cross-site scripting(XSS): It's an injection attack where a malicious script is sent to the server. Its main purpose is to modify page content when data loads into the browser. All the data received at the backend is verified and sanitized to avoid XSS.

4) Hyperledger: All the valid and approved transactions are recorded over the blockchain-based Hyperledger platform. It provides integrity and traceable transactions. Hyperledger includes one organization having three peers and an ordering service following raft consensus.

5) Masking and Hashing: Data Masking to performed to provide a functional alternative without using actual data. For example, contact information, SSN, and account number are served after masking. While some of the user-sensitive data is stored after encryption.

6) Data Verification and validation: All the data received is verified at the backend to filter out the requests from unauthorized actors. Data is validated using regular expressions to avoid SQL injection and to ignore irrelevant data.

III. Results

Following functions are available for the Customers of the Secure banking system:

1) Debit and Credit Funds: A customer can submit a debit/credit request over the system and a bank employee can authorize or decline the request and the customer's account is updated according to the request's status.

2) Transfer fund: Customers can request/approve transfer funds requests and critical transactions need approval from bank employees.

3) Technical Account Access: Bank Admin is provided with the privileges to alter other employee's accounts and view system logs. Admin is responsible for system maintenance.

4) Help and support center: Customers are provided with the interface to book/schedule appointments with bank employees.

The secure banking system is deployed over the Amazon EC2 cloud and is available 24*7 for user access. All the functional requirements mentioned in the required documents like login, fund transfer, issue, and deposit cashier cheques are working efficiently. The response time of any request is similar to the standard response time of any other banking system. The system is mature enough to withstand the user load of approximately 50 users per second. A chatbot is also added to help users with their general queries and to redirect them to the appropriate page. It's implemented using Amazon lex.

A fully functional web application is submitted within the specified timeline and students are provided with a user guide to help them navigate through all the functions of the banking system. Dummy user accounts were shared with all the students to help them with testing. Many other students were assigned to exploit vulnerabilities and perform ethical hacking on our project. A timeline was specified to conduct ethical hacking over the system. Students submitted their report on the vulnerabilities and all the reported vulnerabilities were fixed. A presentation was given on the fixes and solutions to the reported bugs.

IV. Contribution

This project is a reflection of team efforts. I was involved in all the phases of the projects which were mentioned in the introduction. Here is the list of activities I performed individually being a member of this project team:

A. Backend Implementation:

- I have worked on the backend development part of the project. Initially, I developed an automated script to establish the whole database in one go. I have followed the Object-oriented programming (OOP) paradigm by implementing classes and objects structure. Customer class is a blueprint for customer objects and all their related functionalities and similarly for employee class. The server is running using the 'application.py' file where all of the other modules were imported and all the handles were developed to serve as REST APIs.
- Session management is done at the backend using the python library *flask* to handle all the user requests after one-time authorization. All the data received from incoming requests was sanitized to avoid vulnerabilities.
- I have integrated Hyperledger with the banking system to maintain the integrity of transactions happening in the banking system. Hyperledger comprises one organization having three peer nodes and a membership service provider, an ordering node following the raft consensus to maintain transactions.
- In order to make sure of the availability of the application, I have deployed it over Kubernetes using multiple replicas of the banking system in the containerized fashion inside pods.

B. Synchronization:

- Responsible for maintaining a common Github repository for the project and reviewing the code pushed by other team members.
- Scheduling Weekly zoom meetings and chat sessions with the team.
- Performing code review and suggesting/helping group members with previous web app development experiences.

C. Testing

- Actively participated in the testing phase of the project before deployment.
- A common spreadsheet was maintained to log all the reported bugs during all the phases to avoid people logging the same bugs over and over again.
- Solved all the backend-related bugs and responsible for Github code review.

D. Deployment phase

- Responsible for installing SSL certificate on the server to establish secure and encrypted communications.
- Fixing various bugs reported by other students while the web app was still live without compromising the 24*7 availability of the banking system.

V. Lesson Learned

Here are some of the lessons I learned from this course project:

- Planning and synchronizing group efforts.
- Understanding and analyzing the capabilities of group members and utilizing them to get the most out of it.
- Learned to integrate with people from different backgrounds and cultures.
- Introduced to the OTP implementation and installing the SSL certificates on the web applications.
- Making vulnerability reports, user guides, and all other documentation.
- Container orchestration engine implementation using Kubernetes.
- Performed ethical hacking over assigned projects and learned various new techniques like clickjacking.
- Continuous Integration and development of new features of the banking system.
- Handling bugs and loopholes on a live web app.

Acknowledgment

I would like to thank our professor Dr. Stephen S. Yau, TA Atin Singhal, and my four other team members – Miguel Bouza, Saikat Datta, Priyansh Hanish Mandan, and Tanner Cooper – for this opportunity to take part in this group project.

References

[1] Amazon EC2: Amazon cloud services documentation https://aws.amazon.com/api-gateway/y=item.additionalFields.postDateTime&ec2-whats-new.sort-order=desc

[2] Python Documentation: https://docs.python.org/3/

[3] MySQL documentation: https://dev.mysql.com/doc/

[4] Python One-Time password library: https://github.com/pyauth/pyotp

[5] SSL Certification: Secure Socket Layer Certification https://us.norton.com/internetsecurity-how-to-ssl-certificates-what-consumers-need-to-know.html

[6] Kubernetes: Open source container orchestration engine https://kubernetes.io/docs/home