# SECURE BANKING SYSTEM

Rajesh Surana
School of Computing, Informatics and Decision Systems Engineering
Arizona State University
Tempe, US
rajesh.surana@asu.edu

***Abstract*—In this paper, I present design decisions and functionalities for the Secure Banking System (SBS) project. It allows online secure transactions and management of user accounts. It also covers important security features implemented in this project.**

***Index Terms*—secure banking system, security, attacks, vulnerabilities, user account management**

## I. Introduction

Financial websites like banking systems carry out extremely sensitive transactions and hence proved to be attractive target for cyber-attacks. If these systems compromised, damage could be in the order of millions of dollars. Multiple layers of security is needed so that they are impervious to malicious attacks. For the same purpose, we have implemented various security features in this project such as Public Key Infrastructure, virtual keyboard to enter password, single login in a session, SQL Injection Prevention, role based access control, least privilege principle, SSL over HTTP for communication, etc.

Paper starts with the overview and functionalities that our system has, then it moves on to security features that we have implemented. Next, it discusses my contributions to the project in detail and conclusion as an end note.

## II. Overview

SBS has two types of users – external and internal. External users are further divided into Individual Customer and Merchant/Organization who have their checking/saving accounts with the bank. When external user logs in with valid credentials, he can see his balance, debit/credit amount from his own account, transfer money to another user's checking account, download statement, and initiate request to update his personal information.

On the other hand, external users consist of Internal Employee, Manager and System Administrator. Internal users perform operations on behalf of external users and maintain their account. Regular Employee accept or reject transactions initiated by external user. Basically they perform low priority transactions. Manager has all the privileges of that Regular Employee, in addition, he can perform critical transactions and add/modify/delete external users' accounts with proper authorization. System Administrator manages internal users' accounts and their transaction requests. He also provides PII information to government agency on their request and proper authorization.

### *A. Implementation Details*

1. This system software is written in Java (Spring MVC) [1].
2. Apache Tomcat [2] is used for web hosting.
3. MySQL database is used to store data.
4. Some of the security features are implemented using spring and some of them are accomplished using third party product like virtual keyboard.
5. Google's SMTP sever is used to send emails on behalf of system for new user registration and other important transactions.
6. JavaScript should be enabled for application to be fully functional.

### *B. Security features*

#### *1) Secure Socket Layer (SSL)*

It is a technology which established a secure encrypted connection between web server and your browser [3]. So, all the sensitive private data over the communication channel between our banking system and its users is encrypted and hence secure.

*a) Security use-cases:*

- *Man in the middle attack*

  *Attacker intercepts the communication between sender and receiver and acts as receiver for sender and sender for receiver, sitting in the middle. So, he can modify/delete messages before it reaches the receiver [4].*

#### *2) Public Key Infrastructure (PKI)*

Asymmetric-key cryptography consists of two pairs of keys- public and private. Public key is distributed to whoever you want to communicate with securely and private key must be kept secret. Key pair is selected such that content encrypted by one key can be decrypted by another key [5]. In our application, we use PKI at two places – to delete merchant account and to approve/decline pending transactions by internal employee.

*a) Security use-cases:*

- *Authentication*

  *When user encrypts or signs message with his private key, receiver who decrypts the message using public key of sender, can be sure that message came from legitimate claimed sender.*

- *Encryption*

  *When message is encrypted using public key of user, only that user's private key can decrypt it. So, sensitive message intended for particular user cannot be read by any other user.*

- *Non-repudiation*

  *Sender of the message cannot deny that he sent the message if he signs the message with his private key. This evidence can be accounted in court of law.*

*3) One Time Password (OTP)*

It is an alphanumeric string generated automatically to authenticate user for single transaction/session. For SBS, it is valid for particular user for 20 minutes only and then it is automatically deleted from database. OTP is sent to the user via registered email.

*a) Security use-cases:*

- *Authentication*

  *Only authentic user can get that OTP and perform the transaction. Also, even if OTP is compromised it is valid for limited time so of no use to attacker once expired. It provides additional layer of security.*

*4) Captcha*

It is the image containing some alphanumeric string which can only be read by human. Basis for captcha is that distorted text cannot be read and authenticated by bots [6]. We use captcha at the login page of SBS.

*a) Security use-cases:*

- *Dictionary Attack*

  *Trying all words from a dictionary of common words as password to break into the system. With captcha in place, bot cannot do this attack and for human it is considered to be almost impossible to try each possible word as password to hack into the system [6].*

- *Search Engine Bots*

  *Search Engine Bots can be prevented by specifying 'no-index' tag in the html page. But, it does not guarantee that those bots won't enter the website. But, captcha can successfully achieve the purpose [6].*

*5) Virtual Keyboard*

It is a software interface used to enter characters in some sensitive form fields like password, SSN etc. We have used third party virtual keyboard in the form of JQuery.

*a) Security use-cases:*

- *Key-logging*

  *It is a technique to record keystrokes on the machine, covertly without knowledge of user in order to steal sensitive personal information such as password or some confidential email data. These kind of software/hardware can be installed on public machines to steal information of the user. Our SBS is safe from these kind of vulnerabilities as it has virtual keyboard to enter password.*

*6) Role-based Authentication*

It means allowing access to the particular page only if user has appropriate role. For example only System Administrator can read access logs.

*7) Single logging in a session*

SBS doesn't allow logging second time in the same session. This prevents unauthorized access to other user's account.

*8) Account locking after 3 unsuccessful attempts*

User's account is locked for 1 hour if he enters wrong password for 3 times. This prevents CURL attack.

*9) Right click, refresh and back button disabled*

It disables coping of information as well as replaying transactions which are already performed.

*10) SQL injection prevention*

This attack is prevented using Hibernate ORM and parameterized queries to execute SQL statements.

*11) Salted Hashed Password*

Every user's password is concatenated with random salt and then hashed. So, even if system's database is compromised, there is no way to retrieve original password from hashed password.

*12) Front end and backend form validation*

For each transactions like amount transfer we do front end validation using JavaScript and backend validation in the controller before performing transaction.

## III. My contributions

For first 2 weeks I worked on drafting detailed and complete Software Requirement Specification (SRS) document. In the initial phase of project, I developed two simple applications with MySQL database in order to gain complete understanding of the Spring MVC framework. Functionalities that I implemented-

1. PDF generation for account statement

   I referred [7] tutorial to implement PDF generation in our SBS. Most difficult part was to format content in the pdf.
2. Captcha in the login form

   First I used reCaptcha [8] to implement Captcha at the login page. For that I followed [9] tutorial to implement it in our SBS. But, Google's captcha doesn't work with untrusted SSL connection which was given for this project. So we moved to the simple captcha where I created random 6 letters string in the backend, then rendered it in the login page. When user enters the code in the input field, it is validated against one created in the backend. If both matches then controller proceeds with other validations such as username and password.
3. Virtual Keyboard

   Keith wood's keyboard JQuery library [10] is used to add virtual keyboard at the login page of SBS. This insulates the system from key-loggers.
4. Role based authentication for each page in the SBS

   For each URL mapping in the controller I added check, if logged in user has the proper role to access that page.

5. Front end form validations
   I added validations for all the forms in the SBS. For example, phone number cannot have alphabets or address cannot be longer than 200 letters to prevent database exceptions.
6. Debit/Credit for external user
   It is one of the requirement for the project to be able to debit and credit for external user to/from his account. It is as if he is going to the bank and doing transactions in person. As soon as person clicks on the debit/credit button, transaction request is sent to internal employee for approval.
7. Transfer to and from saving account for external user
   When user wants to transfer money from/to his own another account he can do so directly without authorization from bank.
8. Transfer to external checking account for external user
   User has to submit correct OTP in order to complete submit the transaction for approval to bank.
9. Forget password
   On the login page there is a link called forget password which will take user to the next page where user has to give his registered email ID. Once user enters the registered email id, OTP is emailed to him. After successful OTP validation, user can enter new password which will replace his old password.
10. Saving account option for external user
    When user has only checking account, a button is displayed to create saving account. When user clicks on that button it checks if checking account has more than $4000 amount, if yes, then transfer $2000 amount to saving account and set it up.
11. Access log and suspicious activity monitoring
    When user enters login credentials on the login page, system logs in various type of activities such as captcha failed, username failed, password failed or successful login. This information can be used to block user if he enters the wrong password for 3 times in order to tackle brute force attack. This access log is accessible to system administrator only and he can take appropriate measures in case of suspicious activities. It also stores the IP address of client machine which can be used to find source of attack.

## IV. Conclusion

While developing financial websites like banking system, it is very important to consider security from the first stage of development cycle. It may not be possible to add security features afterwards and may costs lots of money and time. Secondly, defense should not just be single layer of security but multiple layers so that even if one layer fails, other layers don’t allow intruders to come in. Lastly, least privilege principle and role based authentication are very important in sensitive applications like banking systems.

## V. New skills, techniques, or knowledge acquired

For this project, I worked from requirement phase creating SRS all the way to testing phase where I tested functionalities as well as possible vulnerabilities of banking systems. Most important thing I learned in this project is essential security principles for any secure banking system to have and how to implement them, possible security attacks that can happen to such websites and how to prevent them. Now, I can work confidently in Spring MVC and Hibernate frameworks. It was very challenging task to properly distribute/coordinate tasks between other team members and integrating it before deployment. I learned great deal of project management skills and handling critical issues with intimidating deadlines.

## VI. Team Members

- Rajesh Surana
- Apte, Tanvi
- Kamath Burde, Varun
- Krishnamurthy, Shankar
- Sangani, Sagar
- Shakthidharan, Mahathi
- Shastry, Aneesh
- Sockalingam, KarthikNarayanan

## Acknowledgment

I thank Professor Yau for such a wonderful opportunity and his valuable guidance for this project.

## References

[1] Spring Framework, [online]. Available: https://spring.io/. Pivotal Software Inc.

[2] Apache Tomcat, [online]. Available: http://tomcat.apache.org/. The Apache Software Foundation.

[3] Secure Socket Layer, [online]. Available: https://info.ssl.com/article.aspx?id=10241.

[4] Man-in-the-middle attack, [online]. Available: https://www.owasp.org/index.php/Man-in-the-middle_attack.

[5] Public Key Infrastructure, [online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/bb427432(v=vs.85).aspx.

[6] CAPTCHA: Telling Humans and Computers Apart Automatically, Carnegie Mellon University, [online]. Available: http://www.captcha.net/.

[7] Spring Web MVC with PDF View Example (using iText 5.x), CodeJava, [online]. Available: http://www.codejava.net/frameworks/spring/spring-web-mvc-with-pdf-view-example-using-itext-5x.

[8] reCaptcha, [online]. Available: https://www.google.com/recaptcha/intro/index.html.

[9] reCAPTCHA and Spring MVC integration, [online]. Available: http://www.codingpedia.org/ama/recaptcha-and-spring-mvc-integration/.

[10] JQuery keypad, [online]. Available: http://keith-wood.name/keypad.html