Computer Networks and Security

Bank-ATM Network and Security Model

GROUP 1

Heer Kubadia (22110096)
Jiya Desai (22110107)
Lavanya (22110130)
Nishi Shah (22110171)
Pratham Sharda (22110203)
Harshita Singh (22110299)

Introduction

This report presents the architecture and working of a Bank-ATM network and security model. The model consists of two programs: the bank (which acts as the server) and the ATM (which acts as the client). The atm program allows customers to interactively perform operations such as account creation, deposits, withdrawals, balance checks and transaction history. The bank server handles the requests received from the atm using a database that stores the relevant information of registered users. The ATM user interface (on the terminal) looks like this:

```
ATM Menu:
1. Register
2. Login
3. Exit
Enter your choice: 1
Enter username (allowed: [_-., digits, lowercase letters], 1-127 chars): nisha
Enter password: shih
Enter initial deposit amount (must be formatted as 0.00): 67.00
Server: Registration successful!
ATM Menu:
1. Register
2. Login
3. Exit
Enter your choice: 2
Enter username (allowed: [_-., digits, lowercase letters], 1-127 chars): nisha
Enter password: shih
Server: Login successful! SessionID: SAUWTVWICWGSDUFS

Logged in. What would you like to do?
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. View past transactions
5. Logout
Enter your choice: 1
Server: Current balance: $67.00
```

User Interface

The ATM user interface (on the terminal) prompts the user to choose one of three options:

1. **Register (for new account creation)**:
    - **Enter Username**: If the username does not meet the specified requirements, the user will be asked to enter a new one and will continue to do so until a valid username is provided.
    - **Enter Password**: The user inputs the password for their account.
    - **Enter Initial Deposit Amount**: The user enters the initial amount to be deposited in the bank. This amount must be greater than $10.00. If not, an appropriate message is displayed on the ATM side, and the user is prompted to select from the three options again.
    - If the username already exists in the bank database, the message “Username already exists” is displayed, and the user is prompted to select from the three options again.
2. **Login (for existing users)**:
    - The server checks the username and password (hash). If they match, the bank generates a session ID using generate_session_id() and stores it in the active_sessions map. Response: "Login successful! SessionID: <id>" or "Invalid username or password!"
      If the username and password do not match, the bank sends the response: “Invalid Username or Password.”

Once a registered user has successfully logged in, a unique **session ID** gets generated, active for the entire duration that the user stays logged in. The user then gets prompted to choose one of the following options:

- **Check Account Balance**: The session ID is sent along with the command to check balance to the bank, and the server responds with the user's current balance.
- **Deposit Money**: The user enters an amount to deposit, and this is sent to the server along with the session ID. The server updates the balance and sends back the new balance. The user input also gets validated before the request is sent to the server (i.e., if the entered value is numeric, non-negative, etc).
- **Withdraw Money**: Similar to deposit, the user enters an amount to withdraw. The bank checks if there are enough funds, then deducts the amount from the balance and sends back the updated balance. The user input also gets validated before the request is sent to the server. If the transaction would result in a negative balance, an appropriate message is displayed on the ATM side, and the transaction is canceled.
- **View past transactions (additional functionality):** The user can check the past transactions made along with the timestamps of the transactions using this option. The bank logs the transaction history of active users in a separate csv file. It retrieves the information from that file upon receiving this request.
- **Logout**: The session is terminated, and the user is prompted again to choose from register, login, and exit.
- **Exit**: This option is used to close the application and exit the program.

Model Functionalities

Our model has the following functionalities corresponding to the requirements of the project:

- **Unique Username and Password**: Each registered user has a unique username and password, stored in the 'card file' of the user. It is assumed that each valid user possesses this information and may share it with others.
- **Auth File**:This is the primary means of authentication between the ATM and the bank. When the bank program starts, the bank generates an authorization file. This file is also present with all authentic clients (ATM) through a secure channel (assumed to have been done already). So when the client tries to connect with the server, it will send its own auth file to the server. If it matches with the server's file, the authentication is successful. This way, only recognised ATM programs will be able to connect to the server, unless the auth file gets compromised. In addition, we have also implemented a mutual authentication process using digital certificates for both the bank and the ATM, issued and verified by a certificate authority (explained in detail later).
- **Functionality Corresponding to Commands**:
    - **-n <balance>** (Create a new account with the given balance): The user can use the option of "Register" to create a new account with an initial deposit amount.
    - **-d <amount>** (Deposit the specified amount): Once a registered user has logged in, they can use the "deposit money" option to do the same in our model.
    - **-w <amount>** (Withdraw the specified amount): Once a registered user has logged in, they can use the "withdraw money" option to do the same in our model.
    - **-g** (Get the current account balance): Once a registered user has logged in, they can use the "check balance" option to do the same in our model.
    - **SIGTERM signal** (Terminate the ATM application): For a user who has an active session going on, they can use the 'logout' option to end the session to go back to the three original options (register, login or exit). Then, selecting 'exit' will terminate the ATM application
    - **-p <port>**: We are using the port 8080 to establish a connection between the bank server and the ATM.

Added functionalities:

- The bank can accept transaction requests from multiple ATMs concurrently. It can even accept the same user login from multiple ATMs while preventing clashes.
- Considering the user factor, we have developed an easy and accessible interface for a better user experience.
- A transaction log has been maintained with the corresponding time stamps specific to each user, which users can view after logging in.
- Passwords are constrained by a set of rules:
    - minimum 8 characters
    - must include uppercase, lowercase, number, special character

  If this condition is not satisfied, then the server will prompt again for a stronger password.

- Not only are we verifying the auth file, but bank and ATM certificates (which have been authenticated by the Certificate Authority) are also used for better security.
- After ten (10) commands have been made by a single user, the user will be logged out, and they must log in again to continue using the ATM. This is to prevent a DOS (Denial of Service) attack.

Error Handling

Protocol Errors

- If the bank fails to respond to the ATM's requests within 10 seconds, the ATM exits with return code 63.
- If the ATM cannot establish a connection with the bank, it exits with return code 63.

Other Errors

Error 255 is triggered in the following scenarios:

- **Login Mismatch**: If the username and password do not match during login, an error message with code 255 is shown, prompting the user to choose again from the options (Register, Login, Exit).
- **Invalid Username**: If the username does not meet specified requirements, an error message with code 255 is displayed, and the user is prompted to enter a new username.
- **Invalid Initial Deposit**: If the initial deposit amount is invalid according to specified requirements, an error message with code 255 is displayed, prompting the user to select from the initial three options again.
- **Authentication Mismatch**: If the ATM's and bank's auth file do not match, the ATM exits with return code 255.
- **Invalid Deposit Amount**: If the deposit amount is invalid according to specified requirements, an error message with code 255 is displayed, prompting the user to choose from the four options (check balance, deposit, withdraw, logout) again.
- **Invalid Withdrawal Amount**: If the withdrawal amount is invalid or exceeds the current account balance, an error message with code 255 is displayed, prompting the user to choose from the four options (check balance, deposit, withdraw, logout) again.
- **Invalid Choice:** If the choice entered by the user is not one of the given choices, an appropriate error message is displayed.

Configuring the ATM-Bank model

Follow these steps to configure the ATM-Bank model:

1. Download the Build Folder: This folder contains the Makefile for macOS and Ubuntu, the C++ codes for the ATM and Bank, and a requirements folder with the necessary files to run the codes successfully.
2. Compile the Code: Use the Makefile to compile atm_final and bank_final to create the executable files atm and bank, respectively.
3. Configure Server IP: Store the IP address of the machine running the Bank in server_ip.txt. Ensure that port 8080 is available for the Bank to operate.
4. Run the Bank: Start the Bank executable with the command: ./bank
5. This will generate an authentication key, which will be stored in bank_auth_file.txt. Copy this key and paste it into atm_auth_file.txt (assumption - this communication/exchange is considered secure).
6. Run the ATM: Now, execute the ATM with the command: ./atm

Voila! You are now in the ATM interface, ready to use the ATM-Bank model.