COP 5615 – Project 4(Part II)

Secured Facebook Simulator

Priyanshu Pandey (UFID: 21081358)
Sourav Kumar Parmar (UFID: 86511933)

Instructions to run the program Build system: Linux (Ubuntu):

- Unzip the **PriyanshuPandey_SouravParmar_Project4.tar.bz2**
- Open Three terminal window
- Change directory to the following
  Terminal 2: Path: PriyanshuPandey_SouravParmar_Project4/SecuredFacebooksimulator/Server
  Terminal 3: Path: PriyanshuPandey_SouravParmar_Project4/SecuredFacebooksimulator/Webserver
  Terminal 1: Path: PriyanshuPandey_SouravParmar_Project4/SecuredFacebooksimulator/Client
- Execute the command sbt run in each of the terminal in following sequence.
  1st Server 2nd Webserver 3rd Client

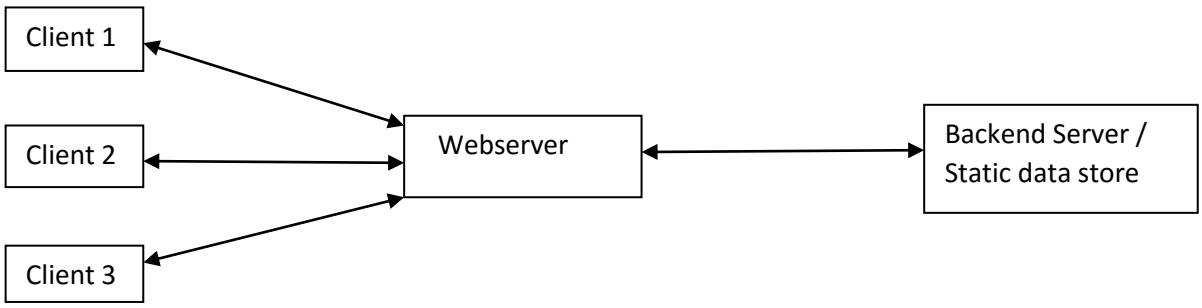
Framework Used : Spray-can, Akka

Overview: Secure Facebook Simulator is enhancement of project 4 part I. It provides end to end encryption system where all data and communications are encrypted at client. Decryption of data is possible only at intended recipients

Encryption Features:

- RSA 1024-bit key cryptography
- 128 bit AES encryption
- Data encryption
- User authentication using FIPS 196

Project Architecture:
Three layered Architecture
1st layer: Client layer – This layer is used to simulate the behaviour of end users.
2nd layer: Webserver: Interface layer between Client and Backend server. This take cares of routing of client request to backend server and vice versa.
3rd layer: Backend Server: This is a database layer that maintains all the datastuctures and handles request from client.



Implementation Details:
Client: Worker actors are generated which simulate client behaviour. Handling for createuser , createpost, pageview ,addprofilepic, addfriend , , addalbum etc are simulated.
Encryption added for createpost, viewpost, addalbumpics, viewalbumpics, userAuthentication

Image is simulated using Byte array.

Encryption Implementation:

1) Data (Image/Post)

- Each post is encrypted with AES key and unique IV.
- AES key user A is encrypted with public key User B
- While viewing data AES of A is decrypted with privatekey of user B
- Generated AES key of A generated above is used by B to view A data

2) Authentication

- Client sends Hello message to Server
- Server generates secure Random number and challenges client to authenticate
- Client encrypt sent random number and certificate with self-private key and notifies server.
- Server checks if sent random number is same it verifies certificate
- Server sends Success or Failure token to client depending on result of above step

Webserver: It accepts the Http request from client and call the respective handlers of backend server. It acknowledges the request with error codes (200 ok , 400 handler not found)

Backend Server: This acts like a static Database store and provides handler to fetch data corresponding to the request from client.
For Post request from client entries are updated in corresponding data structures.
For get request from client corresponding value is fetched if exist and is sent back to client.

Stores Metadata and encrypted data.
Not capable of decrypting client data

Observation:

Server Logs

```
Current Average(of Last 10 sec) = 0
 Max Average Requests PerSecond(Over period of 10 sec)= ***** 2509 ********
```

Client Logs:

```
Encoded Post:
//H3/+A1i8pM6d9UzayWwJ4c1yGE75DBNB/Hv5BcPzQzsnYfM78Ia05akn/PFlTh9giNbNjw11BY/RhAp+SHaltrLPA8fgRwb+lquZDBpj
k=)
Post: In three words I can sum up everything I've learned about life: it goes on.
```

Conclusion:

From observed logs it can be concluded that end to end encryption is implemented.

References:

https://zephoria.com/top-15-valuable-facebook-statistics/
https://github.com/spray/spray-json
http://www.javacodegeeks.com/2014/11/first-steps-with-rest-spray-and-scala.html