SYSTEM AND SOFTWARE DESIGN DESCRIPTION (SSDD): FOR

An Internet Instant Messaging System

Version 1.0
2016-10-23

Prepared by:
James Combs, Joel Seida
University of Texas at Dallas
Dallas, TX

CS4349.001 SSDD

[put program /system name here]
TABLE OF CONTENTS

Section Page

1 INTRODUCTION

The purpose of this document is to describe the architectural components of secure internet instant messaging system and the supported functionalities of each component, security features, threat model, and attacks considered in a complete and concise manner. The internet instant messaging system supports authentication of its users, confidentiality of the messages and data exchanged between users, and verification of the integrity of messages and data exchanged between its users. This document will describe in general how users log into the system, establish sessions with other users, and how the system provides integrity and confidentiality of message transfers between users as well as any other assumption that have been made in the design of the implementation for the internet instant messaging system.

1.1 DOCUMENT OVERVIEW

This subsection shall provide an overview of the organization of this SSDD.

- Section 2 of this document describes the system architecture.
- Section 3 provides a description of the threat model considered for this system.
- Section 4 provides a description of the system design. That is, it described the protocols for communication and authentication between client-client and client-server.

1.2 ASSUMPTIONS

There have been a number of assumptions that have been made during the design of the system referenced in this document.

1. DOS attacks are not relevant.
2. Server database break attacks are not relevant.
3. Identity hiding is not relevant.
4. Clients may be malicious.
5. Clients trust the server's public key.
6. A client may only have a connection with one other client at any time.
7. The server and client have already exchanged username/password information by some outside mechanism.

2 SYSTEM AND SOFTWARE ARCHITECTURE

This section of the document shall describe, with detail, the relationship and functionality of each component in the system. These components, when integrated together as specified within this document, shall implement all functions performed by the system in response to an input or in support of an output as described by the project requirements specification. All components shall: be uniquely identifiable, be well described, have clear responsibilities, and have well described interactions with other dependent components.

2.1 CLIENT

A client consists of identification information such as IP address port for TCP stream socket connection to the server and other clients. Each client will have a shared session key for confidential and integrity protected message transfer with the server and any other clients that it may have a active session with at the time. The client is not responsible for maintaining any long term state for the sake of mobility and security. The client component is not responsible for remembering any username and password pairs the human using the machine may need in order to log into the system.

2.2 SERVER

The server is responsible for providing each client its buddy list, authenticating each client based on the username and password provided by the client, and handling client-client connection requests in a secure manner. The server should store and maintain the buddy list for each client on its behalf as well as maintaining a secure database of client authentication information.

2.3 LOGIN INTERFACE

The login interface is responsible for sending the client authentication information to the server as well as receiving the the shared session key from the server once the client has been authenticated and pass it to the client workstation for communication with the server. The login interface is also responsible for informing the client of invalid authentication information if the server responds with an invalid authentication response.

2.4 SESSION INTERFACE

The session interface is responsible for establishing a client-client session. The session interface will send client-client session connection requests to the server and receive the response from the server. The response consists of a shared client-client session key as well as a ticket to the client being requested for communication. The session interface is then responsible for sending the ticket to the client that has been requested. In this manner, a client-client session may be established. Encryption an authentication keys for communication will be derived from the shared session key received from the server. This will allow confidentiality and integrity of the messages being transferred during the client-client communication session.

2.5 BUDDY LIST

The buddy list is responsible for maintaining client information such as username, IP address, port number for TCP stream socket connection, availability (i.e if a client is online of offline). The buddy list allows a client to have visibility of other clients that it has a direct relationship with. This information will allow clients to create a client-client session request to send to the server. The buddy list is maintained by the server. Clients will send buddy list modification requests to the server in order to make changes to the buddy list it owns. Each client is only allowed to own one buddy list. The server will not accept any modification requests form any clients that do not own the buddy list.

2.6 CLIENT ACCOUNT DATABASE

The client account database is responsible for storing client authentication information such as username, SHA512 hash of password concatenated with a random salt, and the salt that is associated with the password hash. The client account database may only be modified by the server. The server will own only one client account database.

3 THREAT MODEL

This section of the document shall list the different assets that have been identified to be of important to security, the different attacks considered that could harm or compromise the assets, and the security features that are implemented to mitigate or prevent the attacks that could harm or compromise the assets identified.

3.1 ASSETS

The assets identified to be of importance are:

- Username/Password pairs
- Session keys for client-server and client-client communication
- The server's private RSA key

3.2 ATTACKS CONSIDERED

The attacks that have been considered against the assets described above are:

- Eavespropping based attacks
- Session hijacking
- Malicious clients

3.3 SECURITY FEATURES

Security features correspond to the actions taken to mitigate or prevent the attacks on the assets considered above.

- RSA Authenticated Diffie-Helman session key exchange to allow clients to establish a secure session with the server.
- AES encryption in Cipher Block Chaining mode of operation for confidentiality of message transfer.
- CBC Residue for integrity verification of each message being transferred.

4 PROTOCOL DESIGN

4.1 CLIENT-SERVER / LOGGING IN

Step 1 - Initial Exchange (Login):

1. ClientA → S_{RSA}\{uname\} → Server
2. ClientA ← S_{RSA}[N_S , salt] ← Server
3. ClientA →S_{RSA}\{uname, hash(pw, salt), N_S\} → Server
4. Server decrypts and checks database for ClientA information.
5. ClientA ← S_{RSA}[g^s mod p] ← Server
6. ClientA → S_{RSA}\{g^a mod p\} → Server
7. Both agree on $K_{SA} = g^{sa}$ mod p

4.2 RETREIVING BUDDY LISTS

Step 2 - Server Gives ClientA Buddy List

1. K_{SA}\{ClientA buddy list\}
2. ClientA receives his buddy list

4.3 CLIENT-CLIENT SESSION ESTABLISHMENT

Step 3 - ClientA Requests to talk to ClientB

1. ClientA → K_{SA}\{ClientA, "Want to talk to ClientB"\} → Server
2. ClientA ← K_{SA}\{K_{AB} , "ClientB", ticket to ClientB = K_{SB}\{ClientA, K_{AB}\}\} ← Server

4.4 MUTUAL AUTHENTICATION

During mutual authentication, order of the clients in the encrypted message with the shared session key between the clients matters. The first client in the message is the originating client while the second client in the message is the destination client. That means, switching the order can allow authentication of the clients.

Step 4 – ClientA sends ticket to ClientB (Mutual Authentication)

1. ClientA → K_{AB}\{ClientA, ClientB\}, ticket to ClientB → ClientB
2. ClientA ← K_{AB}\{ClientB, ClientA\} ← ClientB

4.5 DERIVING ENCRYPTION/AUTHENTICATION KEYS

Step 5 – Derive Encryption and Decryption keys and well as message authentication keys

1. ClientA computes $E_A = K_{AB} + 1$ for encryption and $D_A = K_{AB} + 2$ for decryption
2. ClientB computers $E_B = K_{AB} + 2$ for encryptions and $D_B = K_{AB} + 1$ for decryption
3. ClientA computes $Auth_A = K_{AB} + 3$
4. ClientB computes $Auth_B = K_{AB} + 4$