Stevechat

Software Requirements Specification

25.3.2015

Pavel Konarik

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification is to provide detailed description of this product, how it works and real-life implementation options. This document is intended for developers and admins who shall implement this software.

1.2 Scope

This document will cover capabilities, structure and features of this product.

1.3 Definitions, Acronyms, and Abbreviations

User - person who will use this product
Application server – Part of this software that is meant to be run on physical server with public IP. Application server handles connections to each client.
Client – Client is second part of this software distributed to every user. Client is meant to connect to application server.
GUI – Graphical user interface

1.4 References

Oracle, “Class Socket” [Online]. Available: http://docs.oracle.com/javase/7/docs/api/java/net/Socket.html. [Accessed March 2015].
Oracle, “Class ServerSocket” [Online]. Available: http://download.java.net/jdk7/archive/b123/docs/api/java/net/ServerSocket.html. [Accessed March 2015].
Oracle, “Generate Public and Private Keys” [Online]. Available: https://docs.oracle.com/javase/tutorial/security/apisign/step2.html. [Accessed March 2015].

1.5 Overview

This Software Requirements Specification will give you an overall view on this application and its potential use.

2. OVERALL DESCRIPTION

2.1 Product Perspective

Purpose of this software is to allow communication between multiple parties simultaneously. In case of eavesdropping software cannot allow other parties to view content of the messages, therefore implementation of asymmetric cryptography is necessary.

Software allows users to connect either via localhost or via the Internet. For user convenience, software is split into two independent executable files.

First one being an application server. This application server handles list of online users, login and registration process, forwarding messages to different users and other requests from clients. Solution that includes external server with public IP is the most user-friendly approach. Without a server, to communicate via the Internet, each user would have to run client and application server on their computer, therefore they would have to forward certain ports on router to allow other clients to access their application server. Letting company to run the application server on their machines (probably servers) with public IP eliminates this major issue. Since application server is meant to be run and managed remotely via terminal or shell it does not need graphical user interface.

Second executable is a client. Client provides GUI for easy communication with server. It will construct message packets that will be interpreted by the server. Packet structure is consisted of exclamation mark followed by a command name and arguments. For example login packet would be "!login username password". Client takes input from user, construct packet, sends it to the server and listens for answer. On every successful login client generates new public and private key. Other clients can send a request key packets and public key exponent and modulus will be send back to them.

2.2 Product Functions

Application server is capable of running on either localhost or IP provided during startup. Port is also modular and can be modified as a variable during server execution. For example root@stevechat:~# java -jar SteveChatServer.jar 5468 93.185.105.58 will start server on port 5468 listening on IP 93.185.105.58. Server only knows who is communicating with who, but it cannot read the message itself since it is encrypted. Typical communication should look like this "!msg Pavel Peter
hSwC/xsimdDomaYK9+fCR24uzrfMq5C4j2pYeh4//72ds2sV7+lf7eOkYCu5UWSZHuHwODS
TBS5W5sNIdPOyuDe505aFaE4is2uUm3xEgMAYn0GeFmnqe1MGn70drPiVqStHvzYcAhDd
MRxDdbCOS4NsvlbyxTicuvWwF6pSScPvsyAp6p79j4IfeYASUB8g0C4PtHy4zcunZ4Wa1RRI
j3YN49y639yIxF+QqZdcZ4dsqj3Tv8NfGvVijv/6zyClH7fUylG0s2SBizLIpQkIfRBWURBoaEn
hqO9MvTjUeNSMhoXK1Lqss9pJL3U5w33d+uOOlTlMF2lN5k96ju3VvQ==". We can determine that Pavel has sent Peter a message, but without private key we cannot decrypt the message. Default key length is set to 2048 with PKCS1Padding, but can be edited in class containing other general variables.

Client is split into multiple layout cards. On opening the application, user is prompt to either login or register new user. A small combo box should allow user to choose different server and ports. This might be used by company to create backup servers or localhost servers for communication during meetings, where users are on the same local network.
After pressing login button, special packet with username and password is sent to the server. Server verifies these information and sends appropriate response.
If login was successful, client GUI card is changed and user sees other online users in list on the left side of GUI. At this point new public and private key is generated. By clicking on one user in the list, chat is set as enabled and user can send first message. Before the first message is sent,

clients exchanges their public keys, so the communication between them is encrypted. After user closes their client or new client joins, every other client refreshes their client lists. Also every inordinate situation is handled. For example if server crashes while multiple people clients are online, they are logged out, informed that server is down and prompted to try logging in after few minutes.

2.3 User Characteristics

This product is meant for a company that has to consult and share secrets between employees via the Internet and is afraid of eavesdropping or man-in-the-middle attacks. However, public release for everyone who wants to communicate securely via the Internet is also possible. Client is very user friendly and requires no previous computer skills.

2.4 General Constraints

Since this project is developed in Java, it is cross platform application. On the other hand, Java has to be installed on that machine.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Server shall not include user interface.
Client shall include interface with two cards. First card shall allow user to login, register, change server and port. Second card should provide list of other online users, text area where chat communication is displayed and field for writing message.

3.1.2 Hardware Interfaces

Both server and client application shall run on machine with ARM1176JZF-S 700MHz processor and 256 megabytes of RAM (Raspberry Pi 1 model B+) and better.

3.1.3 Software Interfaces

This application shall be able to be run on any machine with Java 1.7.0_67 and higher.

3.2 Functional Requirements

3.2.1 Server

3.2.1.1 General

I. Server does not contain GUI.
II. For every client server generates new threat that listens for and handles messages from this user.
III. Server keeps list of online users and sends an update message to all online users if the list changes

3.2.1.2 Inputs

I. When started, server accepts two input variables, port number and IP that server runs on (can be localhost).

II. (Optional) Server should check if variables are in the right format.
III. Server creates new thread that listens for new client on the start of an application server or every time new user joins.
IV. When user connects, listening thread becomes personal thread for this user. Only this tread can listen to messages from this user.
V. If user disconnects, end his listening thread.

3.2.1.3 Processing

I. Messages from clients are processed based on message command type.
Server must handle these command types in particular way-
   a. !register username password – Server checks if user with same username is not in our database and if username us unique, server adds new user to the database and sends positive response back to client. If username is already in the database, server sends back negative response and terminates listening thread for this user.

   b. !login username password – Server checks if username and password is in our database and if both arguments matches the database, login was successful and server sends back positive response, otherwise login request is rejected.

   c. !msg arg0 arg1 arg2 – Server forwards message (arg2) from user arg0 to user arg1.

   d. !key arg0 arg1 arg2 arg3 – Server forwards key modulus (arg2) a public exponent (arg3) from user arg0 to user arg2

   e. !requestKey arg0 arg1 – Server forwards request for public key from user arg0 to user arg1

3.2.1.4 Outputs

I. Server sends message "!system Loggedin" on successful login or registration.
II. Server sends "!system Wrong username or password" when received wrong login information.
III. Server responds with "!system User already registered" when user tries to register same user multiple times.
IV. Server forwards from one user to another encrypted messages, public key requests and public key information.
V. On any change to list of online users server sends "!onlineList arg0 arg1..." message to all clients updating their online lists.

3.2.2 Client

3.2.2.1 General

I. Client must have GUI
II. GUI shall have two cards
III. First GUI card shall include registration and login forms and combo box for different predetermined servers.
IV. (Optional) First GUI card should also allow user to change port
V. Client must allow user to login or register
VI. Users must have unique usernames
VII. One user cannot be logged with multiple instances
VIII. Client must allow multiple one (user) to one (user) conversations
IX. Client must handle all system messages listed previously in this document from the server
X. Client must have list of other online users
XI. (Optional) Client must change background color of user in the list, if it receives a message from that user
XII. Client must use asymmetric encryption to encrypt and decrypt messages
XIII. Client must allow user to write and send messages to other user

3.2.3 Encryption

3.2.3.1 General

I. System must use asymmetric encryption (RSA).

3.2.3.2 Processing

II. Encryption object shall store private and public key.
III. This class shall encrypt and decrypt raw text.
IV. On every successful login new pair of public and private keys shall be generated.

3.2.3.3 Outputs

I. This encryption handler object shall be able to send and request public keys.

3.5 Non-Functional Requirements

3.5.1 Performance

Login process shall take less than a second for computer running Windows 7 64bit with processor i7-4770K at 3.50GHz with 16 GB of RAM with stable internet connection.

3.5.2 Reliability

I. Public server must have weekly up-time of 98%

3.5.3 Availability

I. Public server must have public IP and ports application uses forwarded to it.

3.5.4 Security

I. Server has to store login credentials in protected file. Policies has to be set that only authorized users can gain access to this file.

II. People eavesdropping on conversation or running the server cannot read the raw (decrypted) messages.