Software Design Specifications

Version 0.1

December 8, 2015

TauNet

Submitted in the partial fulfillment of the requirements of CS 300 Software Engineering

Table of Contents

1.0) Introduction

The purpose of this document is to provide a description of TauNet and its structure and design. It is designed for individuals involved in the development of TauNet.

1.1) Goals and Objectives

TauNet is designed for the raspberry pi and will allow for any user to send SMS messages (i.e. text messages) to another user over the internet securely. TauNet will use the RC4 encryption to send message securely. In addition, all users of TauNet need to have the same key for encryption and decryption. To establish a network, the TCP protocol will be used; this protocol is broken down into two subsections: the client (sender) and server (recipient).

1.2) Project Overview

TauNet is composed of two primary components: a client-side application and server-side application. In addition, there are two groups of features: core features and additional features. Core features are essential for the system to function while additional features are to add extra functionality. All implementations of TauNet are written in Python.

1.3) Core Features

1) Main Menu
    - **a.** Allows user to select from a list of options and control the program flow
2) CipherSaber 2
    - **a.** RC4 encryption, decryption, and key scheduling are used to send messages securely.
    - **b.** Encryption is automatically done when user is done writing message and decryption is automatically done when user receives a message
3) TCP
    - **a.** Method of sending messages between the clients in the network

1.4) Additional Features

1) Contacts/Address Book
    a. All contacts are saved in a CSV file and then uploaded to the program when the program starts
    b. View, add, and delete contacts from the Address Book

1.5) Address Book Data Structure

The data structure for the address book is a linear linked list. Each node will contain one contact along with that contact's information (name, IP address, and port number). The IP address and port number will be hidden from view. However, when adding a new contact, the IP address and port number of that new contact needs to be known in order to send messages successfully.

1.6) References

1) TauNet Communications Protocol v0.2 (revision 1)

2) Software Design Document Example:

https://cise.ufl.edu/class/cen3031fa11/documents/examples/SDD_Example_1_2011.pdf

3) TCP Server Code that was modified for TauNet:

http://stackoverflow.com/questions/17453212/multi-threaded-tcp-server-in-python

4) Basic TCP Client and Server Code used as framework:

https://wiki.python.org/moin/TcpCommunication

2.0) Architecture and Component Level Design

As mentioned earlier, TauNet is composed of two primary components: a client side application and a server side application.

2.1) Client

The client side application allows the user to write a message and then send it to a recipient. It is composed of the following three functions:

1) Client: This function connects to the recipient, formats the message as required by the TauNet protocol, and closes the socket connection.
2) Multi-Line Message: This function allows the user to write a message that is greater than one line. It allows formats the message to match the TauNet protocol.
3) Client Main: This function creates the socket object and then calls the Client function with the appropriate arguments.

2.2) Server

The server side application allows multiple clients to connect and send message, store all messages received to a list. The server uses daemon threads in a way that one daemon thread is assigned to one client that is connection. The reason for daemon threads over regular threads is because daemon threads don't interfere with the exit of the program. When the server is run alone, it will run forever till the user enters CONTROL –C to exit. When the main program is executed, the server runs in the background till the program is terminated. The server application is composed of a function and a server:

1) Server Main: This function creates socket object, sets the port number to a specific port and keeps all threads created in a threads list which is then killed once program is finished running.
2) Server: This class initializes the thread object and the socket and client address parameters. It also has a run function which tells the class to run this function immediately after initializing. The run function accepts the connection and receives the data as well collecting the date and time when the message for received.

2.3) CipherSaber 2

CipherSaber 2 is used as the cryptography protocol. It uses RC4 to send messages securely to other clients and used to decrypt encrypted messages received from other clients in the network. The client side of the program calls upon the RC4 encryption function, while the server calls the RC4 decryption function. The RC4 follows the pseudocode given by Bart Massey (https://github.com/BartMassey/ciphersaber2) and consists of three functions:

1) RC4: This function produces the keystream used in the encryption and decryption. It is dependent on specific rounds of key scheduling (20 for TauNet) and a given key that all clients need in order to properly encrypt and decrypt the message.
2) RC4 Encryption: This function returns the message given in an encrypted form. It calls the RC4 function within it and is dependent on a key.
3) RC4 Decryption: This function returns the message in its decrypted form. It calls the RC4 function within in and is dependent on a key.

3.0) User Interface

The user interface for TauNet is a menu interface. This interface controls the program flow and consists of 8 options as shown below. Each of the options is explained in the table following the image below.

Figure 1: Menu Interface

Table 1: Menu Option Description

Menu Option	Description
View Messages (Current Session)	Allows the user to see all messages received since the program was started, but not messages from previous sessions
Send Messages	Allows the user to send a message to a specific client(s)
View Message History (Previous Sessions)	Allows the user to see all messages received from previous sessions
Delete Message History (Previous Sessions)	Allows the user to delete previous sessions message history
View Contacts	Display all your contacts names in your address book
Add Contact	Add a contact(s) to the address book
Remove Contact	Remove a contact(s) from the address book
Quit	Exit the program