Software Design Document

Version 1.0

December 7, 2015

TauNet System

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

Table of Contents

1.0. Introduction

1.1. Purpose

The purpose of this document is to describe the architect and system design of the TauNet software system. It will explain how the system will be implemented to meet the requirements outlined in the TauNet SRS document. This document is intended as a primary reference for the developer of the TauNet software system.

1.2. Scope

The TauNet software system will be a secure method to communicate with friends through the internet. The system will be designed to allow two users to send encrypted text messages to one another over a standard internet connection. It will be designed to run on specific hardware, namely the Raspberry Pi Model B2 single-board computer. This document describes how this system will be implemented.

1.3. Reference Material

TauNet Documentation:

- S*oftware Requirements Specification Document for TauNet system, by Luke Kazmierowicz*
- *TauNet Communication Protocol Document*

References for formatting this document:

- IEEE. *IEEE Std 10161998*
- *Software Design Document (SDD) Template, by Concordia University*

2.0. System Overview

The TauNet software system will be implemented using the programming language Java and designed to run on Linux. It will use an OO (object oriented) design with modules to perform each separate task. The next section will go into more detail about each objects job and there place in the larger system.

3.0. System Architecture

This section will explain the program structure and relationships between modules. The software will contain four main modules that will work together with smaller objects to send, receive, and encrypt/decrypt messages on the TauNet network.

3.1. Primary Module

The primary module will manage the secondary modules. The primary module is responsible for presenting the user interface and managing the control flow of the separate threads. The primary module will also be responsible for loading the encrypt/decrypt key, system username and contacts from a file into memory.

3.2. Secondary Modules

1. *Encryption object:*

   Uses a CipherSaber2 encryption algorithm to encrypt and decrypt arrays of bytes.

2. *Sender object:*

   Takes a Message object uses the Encryption object to encrypt it, and sends it to the intended recipient.

3. *Listener object:*

On a separate thread the listener will constantly be listening for messages and storing them in a queue to be displayed later.

4. *DisplayMessages object:*

   On a separate thread every second it will check if the queue has any messages in it. If the queue contains messages, it will decrypt them using the Encryption object and display them in the order they were received.

3.3. Data Grouping Objects

In addition to the four secondary objects there will be two more that provide the ability to store users contacts and group data about a message into a single entity.

1. *Contact object:*

   Contains a users username and IP address along with methods of displaying and retrieving that information.

2. *Message object:*

   Contains important information about a message including a Contact object to store the senders information, message version number, intended recipient, and the body of the message.

4.0. Data Design

4.1. Message Queue

Messages received while the user is interacting with the system will not be immediately displayed to the screen. They will instead be added to a queue to be displayed later. The message queue will be a static field in the Primary Module so that it is shared between the message Listener and Displayer objects that run on separate threads.

4.2. User Data

The TauNet software will require there to exist a text file named "*data.txt"* containing the universal encrypt/decrypt key, the systems username, and a list of usernames and their corresponding IP address or fully domain-qualified hostname. The file must be formatted line by line as follows.

1. On the first line there will be a valid encrypt/decrypt key.
2. The second line will contain the username of this TauNet system.
3. The third line is blank.
4. The following pairs of lines will contain a username and then an IP Address

Example file:

```
password
MyUserName

profBart
barton.cs.pdx.edu
johnyApple
198.24.43.107
...
```

On launch this file will be loaded and stored in the Primary Module. The password will be stored in a byte array and the username will be stored in a java string object. The list of usernames and their corresponding addresses will be loaded into Contact objects and stored in a Set data structure.

5.0. Human Interface Design

5.1 Overview of User Interface

The TauNet software user interface will be entirely within the command line. When the program first launches the following options will be printed on the screen:

```
- Enter 'Q' to quit.
- Enter 'C' anytime to compose a message.
- Enter 'R' to reply to the last received message.
Waiting for incoming messages…
```

- When the user enters ‘Q’ the program will immediately quit.
- When the user enters ‘C’ all their contacts and a prompt will be printed on the screen:

```
Authorized TauNet users:
   Username: profBart
   Username: johnyApple
   Username: joeShmo
Enter the username you wish to send a message to:
```

  Until the user enters a username that is in the list they will be presented with the following message:

```
That user is not in your contact. Try again:
```

  If the TauNet system for the username they enter is not online, they will get the following message:

```
The user "SomeUsername" is currently unavailable. Try again later.
```

  Once they enter a user in their contacts that is available, they will get the following prompt:

```
Enter your message:
```

  They will then type their message and hit enter. When the message is sent they will see the following:

```
Your message was successfully transmitted.
Waiting for incoming messages...
```

- When the user enters ‘R’ to reply to a previously received message, one of three things will happen.

  If they have not received a message during this session yet, the following alert will be printed on the screen:

```
There is no previous message to reply to! Enter 'C' to compose a new message.
```

If they have received at least one message during this session they will get the following prompt to reply to the sender of the last received message:

```
Enter your reply to user "SomeUsername":
```

If the last message they received was from a username that is not in their contacts they will get the following error:

```
Can't reply to the message because the user is not in your
contacts.
```

5.2 Receiving Messages

When the receiver thread receives a 0 length message it will immediately discard it. When a message of length greater than 0 is received by the receiver thread it will be added to the message queue. The message displaying thread will print all messages in the queue once every second.

In order to prevent messages from being displayed to the screen while the user is composing a message or otherwise interacting with the system, the message displaying thread will be paused during these times. Once the user interaction has completed the message displaying thread will be un-paused and it will print whatever messages have accumulated in the queue since it was paused.