Project Evaluation Report

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.

2.0) Evaluation

2.1) What I did?

What I did in this project was design and implement a version of TauNet that’s functional but not perfect. Details about what TauNet is explained in Sections 1.1 and 1.2

2.2) How it was done?

This project was done by first researching the basic syntax of Python, since I had no experience in programming in this language. Then, I researched TCP protocols: what TCP was, the library important to implement it in Python, and examples of implementations online. Once I had an idea of what to do, I started designing on how to implement TauNet, and then implementing it. As various features were finished implemented, I ran them through the test plan to see if they were working or not.

2.3) What I learned?

I learned a lot throughout this project. I learned a new programming language, Python. I learned how to do networking through TCP and threading. I learned a cryptography protocol via RC4.

2.4) What I would like to do that I didn't?

One thing that I would have liked to do but didn't was find a way to save message history from previous sessions in an encrypted way. I tried using RC4 when writing to the file, but then errors occurred when reading from the file. These errors were based on the type of reading mode I was in. If I had more time, I would have tried to resolve this issue so that I could have implemented it to enhance the security of TauNet.

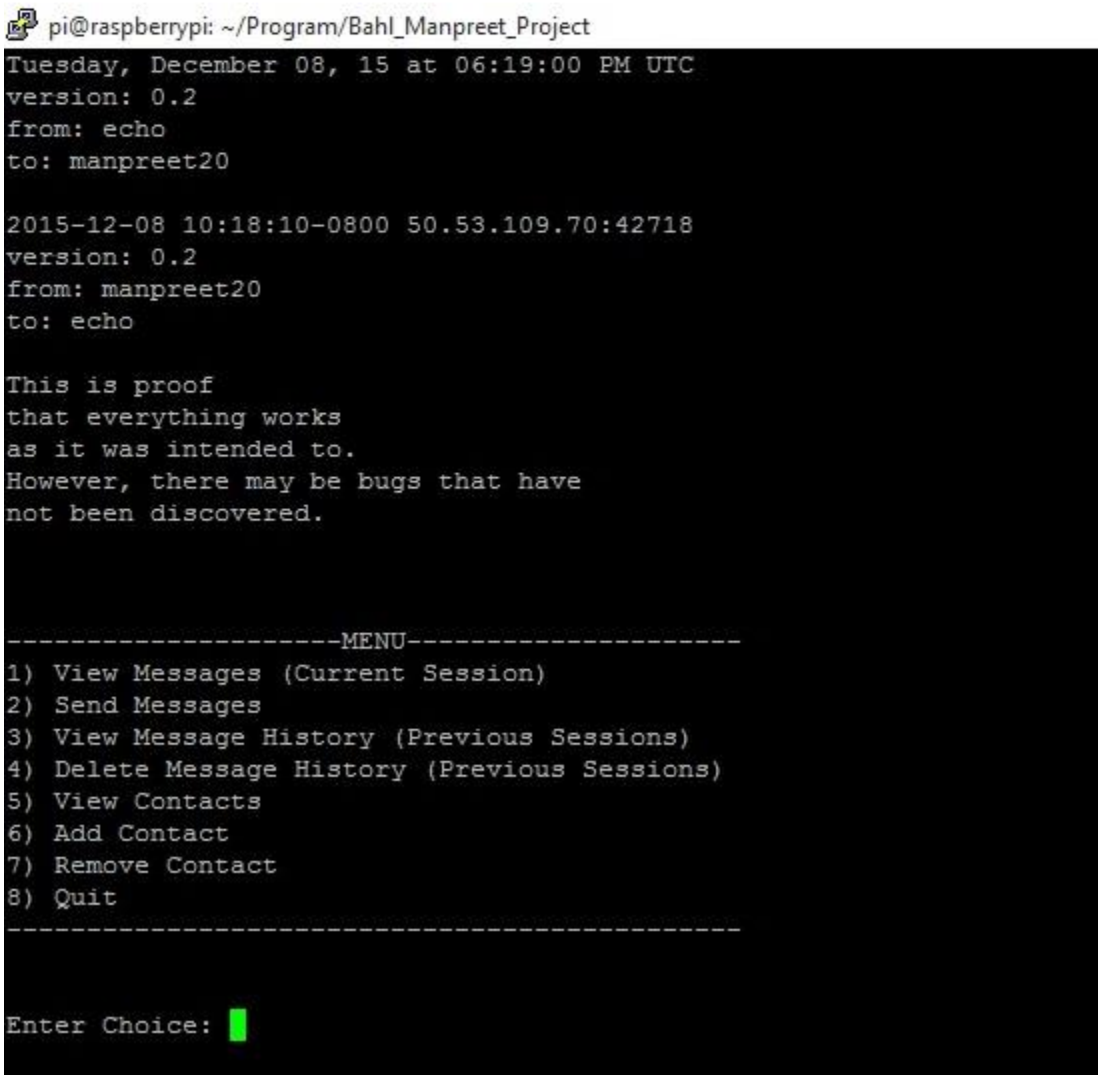
3.0) Test Plan Results

Feature	Test Plan	Result
User Interface	Allow other developers, colleagues, and users run the program and test all the features of the menu. All testers will be encouraged to enter wrong input whenever they choose as to test the error handling of the user interface.	**Pass**
TCP Client	Attempt to connect it with a server implemented by other TauNet developers and the echo server implemented by Bart Massey (Reference 1).	**Pass**
TCP Server	Have other TauNet developers attempt to connect to the server. An additional plan is to use the echo server implemented by Bart Massey	**Pass**
RC4 Encryption and Decryption	Test for RC4 encryption is to have the encrypted data displayed in the same format as the CipherSaber 2 implemented by Bart Massey in Haskell. Test for RC4 decryption is to decrypt the data and see if the message obtained was the same as sent. The echo server implemented by Bart Massey (Reference 1) will also be used to test whether the RC4 implemented	**Pass**
Linear Linked List(Address Book) Features	The test plan for the Linear Linked List features is to implement each feature and then test them thoroughly. **Add:** The plan is to add a contact to the list in alphabetical order. This will be verified after adding a node and then displaying the linked list. **Remove:** The plan is to remove a contact from the list and then display	**Pass**

	the linked list to check whether the contact was removed or not. **Display:** The plan is to test whether the linked list is being displayed correctly by utilizing the add feature and adding some contacts to the linked list in order to test the linked list. **Search:** The plan is to give the feature a name to search the linked list and then display, with a test main, the contact name, IP, and port number. Note, that when testing is done, search will not display the contact's IP and port number, but instead pass them to other functions that depend on the information **Count:** The plan is to have a linear linked list with a known number of nodes and then test this feature by using a test main and display the number of nodes in the linked list. If the number matches, then the feature is working. Various numbers of nodes will be tested to ensure that everything is working	
Loading Contacts to Linear Linked List from CSV File	Utilize the linear linked list display feature and add feature. First, the data from the CSV file will be added to the linked list using the add feature. The display feature will be used to verify if the data was read in properly by displaying all the information.	**Pass**

4.0) TauNet Working Proof

The image on the next page shows the response that is received from Bart's echo server. This shows that TauNet is working properly per the requirements.



pi@raspberrypi: ~/Program/Bahl_Manpreet_Project

```
Tuesday, December 08, 15 at 06:19:00 PM UTC
version: 0.2
from: echo
to: manpreet20

2015-12-08 10:18:10-0800 50.53.109.70:42718
version: 0.2
from: manpreet20
to: echo

This is proof
that everything works
as it was intended to.
However, there may be bugs that have
not been discovered.



---------------------MENU---------------------
1) View Messages (Current Session)
2) Send Messages
3) View Message History (Previous Sessions)
4) Delete Message History (Previous Sessions)
5) View Contacts
6) Add Contact
7) Remove Contact
8) Quit
----------------------------------------------


Enter Choice: 
```