Taunet System Requirement Specification

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to outline the results of requirements elicitation and the analysis of those requirements. This document describes the system in both functional and nonfunctional requirements and serves as a contractual basis between the customer and the developer. The specifications will attempt to adhere to the language of the customer's expertise, avoiding unnecessarily technical terminology.

1.2 Audience

This document is intended for the customer, but users and developers may make use of it when necessary.

1.3 Introduction

1.3.1 Purpose of the System

The purpose of TauNet is to provide a secure, anonymous channel of communication between specific users by providing a method of encryption for messages sent over general Internet traffic.

1.3.2 Scope of the System

TauNet will run on Raspberry Pis using the Raspbian operating system, and support networks of at least 12 users and up to 300 users.

1.3.3 Objectives and Success Criteria

This system will be considered complete once the following criteria are completed:

- A predefined table of users can be loaded into the system with data in accordance with the TauNet protocol listed in the reference section.
- A user can display this predefined table of other users.
- A user can send and receive secure messages to or from any other user listed in the table. This is accomplished using a predefined key in conjunction with the TauNet protocol listed in the reference section.

1.4 References

TauNet Protocol:
https://docs.google.com/document/d/1juKX1KE8FnpVBpb9S6RHwLm2DpCJv0RnuKHOfOK-h7Y

Github Repository of Supplemental Documents:
https://github.com/PSU-CS-300-Fall2015/Tighe_Matthew_cs300Project

2. Proposed System

2.1 Overview

TauNet is a new system that will allow Raspberry Pi owners running the Raspbian operating system to exchange private and secure messages. Users will be able to encrypt and decrypt these messages once they have access to a predefined key. This encryption process is detailed in a common security protocol which can be found in the reference section.

2.2 Functional Requirements

2.2.1 Client Program Requirements

- **Enter a key**
  - A user must enter a key every time the system is started. No other actions can be taken until this key is entered.
- **Menu presented at startup containing options:**
  - Message a user
  - Exit TauNet client
- **Display user table**
  - This will display a predefined table of users listed in accordance with the Taunet protocol found in the reference section.
- **Message a user**
  - The user will be prompted to select a username from the user table, which will be display on this option being selected. Once selected, the user will be able to enter a message of length up to 1 kilobyte. When finished, the message will be sent.
- **Exit TauNet**
  - This option will exit the TauNet client program.

2.2.2 Server Program Requirements

- **Enter Key**

        - The user will be forced to enter a key in order to access other elements of the program.
    - **Receive Message**
        - The server will listen for messages as long as it is running. When one is received, it will alert the user and display the message.
        - On blank messages, a message will be displayed alerting the user that that message has been discarded.

2.2.3 Securing the System

- **Entering a key as first action**
    - This is to ensure that a user has a viable method of encryption.
    - The key will be checked against a hardcoded value to ensure that a user cannot access the user list without it.

2.2.4 Messaging System

- **Message Encryption**
    - Once a message has been entered by a sender, it will be encrypted and delivered to the receiver according to the TauNet protocol listed above.
- **Message Decryption**
    - Once a message has been received, it will automatically be decrypted according to the TauNet protocol listed above and then stored.
- **Automatic Message Deletion**
    - Messages will not be stored.
- **Automatic Message Routing**
    - The receiving location of the a message sent by a user will be determined by the address that corresponds to the user name selected by the sending user.

2.2.5 Downloading the user list

The user list must be located in the same directory as the client program.

2.3 Nonfunctional Requirements

2.3.1 Platform

TauNet will be designed to operate on Raspberry Pi 2B hardware using the Raspbian operating system. This hardware has the following capabilities:

1GB Ram
900 mHz quad-core processor
At least 8GB flash memory

2.3.2 Reliability

The system will perform the same actions the same way during every use.

2.3.3 Performance

- **In-system navigation**
  - Navigating between menus will take within 1 second of selecting a new menu.
- **Displaying messages**
  - Messages lists will be displayed within 1 second of selecting the option.
  - Specific messages will be displayed within 1 second of selecting the option.
- **Displaying user list**
  - The formatted user list will display as soon as the user selects the option to send a message.

2.3.3 Size of system

- **Installation size**
  - Installation of the system will require no more than 10 megabytes.
- **User table storage size**
  - The user table will be as large as the user desires

2.3.4 Message Delivery

- **Latency**
  - A message should be delivered within 5 seconds of completion of the message.
- **Security**
  - Users will also not be able to access the “message a user” function until the key has been checked.
  - Users will not be able to read their received message until their key has been checked.

2.3.5 Implementation

- **Programming language**
  - TauNet will be implemented using the Python language.
- **Allowed characters**
  - All characters in usernames, keys, messages, etc. will adhere to the Unicode Transformation Format-8 standard. This is a standard of character representation that uses 8 bit blocks to represent a character.

2.3.6 Interface

TauNet will be presented within the command line terminal of the Raspberry Pi.

2.4 System Models

2.4.1 Use Cases

Use Case 1

- **Name: User messages another user**
- **Actors: Primary user, client system, secondary user**
- **Description:**

A user opens the client system. They are prompted to enter a key. The user enters a predefined key. The system checks the key and either allows access if it matches or does not allow further access if not. The user then selects the option to message a user.The user list is displayed. The user selects a secondary user to message and is prompted to enter their message. After the message is entered, it is encrypted and sent.

Use Case 2

- **Name: User receives a message**
- **Actors: Primary user, server system, secondary user**
- **Description:**

User opens the server system and authenticates themselves with their key. The server begins listening for messages. When one is received from a secondary user, it alerts the user, decrypts the message, and displays it.