TauNet

Software Design Document

R.J. Russell
12-3-2015

TABLE OF CONTENTS

1 INTRODUCTION

1.1 PURPOSE

The purpose of this document is to provide an overview of the TauNet software design implementation as described in the TauNet Specifications Requirements.

1.2 SCOPE

The goal of this project design is to implement a secured distributed messaging software. This software will enable the user to send messages to other users connected to TauNet. This software is divided into two major components: Client and Server. Each component will be discussed in the section labeled 'Design Overview'.

2 DESIGN OVERVIEW

2.1 DESCRIPTION OF PROBLEM

Communication is a vital aspect of everyday life. As technology increases, our ability to communicate securely through the internet is decreasing. The TauNet software provides reasonably secure communication from one TauNet system to another by using CipherSaber2 encryption algorithms inside an invite only network.

2.2 SOFTWARE OVERVIEW

This TauNet software implementation has two distinct components: Client and Server. This means that it requires two terminals to run this program. One terminal will run the server by executing the tau_server.py script, and will listen for incoming messages. The other terminal will run the tau_interface.py script, which is the user interface and uses the tau_client.py file to send messages to other TauNet users. The following sections provide a break down of each component. Both components use the rc4.py file for encryption and decryption.

2.3 ABBREVIATIONS AND DEFINITIONS

TauNet SRS	TauNet Software Requirement Specification
TauNet Protocol	Document specifying the protocol
TauUser	Another person, other than the user, on the TauNetwork.

3 SOFTWARE IMPLEMENTATIONS

This section provides a breakdown of each file implemented in the software design. The following is a description of the RC4 encryption which is shared with the client and server implementations as well as the two major components, client and server, including any related subcomponents in the software design.

3.1 RC4

This software uses the CipherSaber2 encryption algorithm. The key scheduling is ran 20 times for this particular implementation and a pre-specified key as per the TauNet protocol. All outgoing messages, including header and timestamp, are sent to the RC4 *encryption* method before being sent to the receiving TauNet user. The encryption method generates a random IV, appends it to the key, and sends through the rc4 method to be encrypted. The IV is then prepended to the encrypted message and returned to the client so it can be sent to the intended recipient.

All incoming messages are sent to the RC4 *decryption*, and a timestamp is concatenated to the resulting plain text message before being displayed. The header data is formatted in compliance with the TauNet protocol. The decryption method removes the IV from the message and appends it to the key. Then parses the message from the encrypted message sent in. The message and key are then sent to the rc4 method to so that the message can be decrypted and returned to the server to be displayed.

3.2 SERVER COMPONENT

The server is a stand-alone component that simply listens, decrypts, appends a timestamp and displays the data sent in. The server is in compliance with the TauNet protocol and listens on connects to the required port to operate on the TauNet network. Once the server script is started, it binds to the required port.

When a message is sent to the server, it will read in the message, send it to the decrypt method in the rc4.py file, append a time stamp and then display the message. The incoming message can be up to 1024kb in size. The user will be prompted after each message of how to disconnect from the server, and once disconnected will inform the user that the server has been disconnected. If the server script is not running, no messages can be received. Pressing Control + c will disconnect the server with the correct disconnect handling in place as to not receive error messages.

3.3 CLIENT SYSTEM

The client system is a stand-alone component that is comprised of two sub-components. The Client and the Client Interface. These two sub-components work together to allow the user to enter a recipient, enter a message, encrypt the message, connect to the recipients address and sent the message. The following is a breakdown of each sub-component.

3.3.1 Client Interface

The client interface is responsible for user/client relations. The interface design utilizes the linux command line as per the TauNet SRS. The client interface is a simple design that prompts the user for a receiver, then for a message, and displays each step as the client connects, sends and closes the socket. The design includes displaying the sockets calls so that the user can know the progress of the message through the informative display. The interface also includes a help menu flag and a flag to display the address book, as well as the option to clear the screen with "clear" to give the client interface a better linux command line feel. The client can also exit the program from either the recipient prompt, or the message prompt.

The client interface also is responsible for the address book. The address book is a JSON dictionary. The client interface reads it in, and then compares the users input to the address book. If there is not a match, then the user is prompted with the appropriate messages.

3.3.2 Client

The client component is responsible for making the connection to the recipients address and port. If the connection cannot be made an exception is thrown, a message is displayed and the client interface re-prompts for a recipient. When a connection is established the message is encrypted and then sent and the socket connection is closed to that recipient.

The client and client interface work together to enable sending encrypted messages quickly and efficiently.

3.4 DESIGN DIAGRAM

The following simple diagram shows the path of a message.

- From the client interface the TauUser address and message is received, sent to the client, encrypted, and sent from the client to the TauUser.
- From the a TauUser on the network, a message is received by the server, decrypted, and displayed in the localhost terminal.

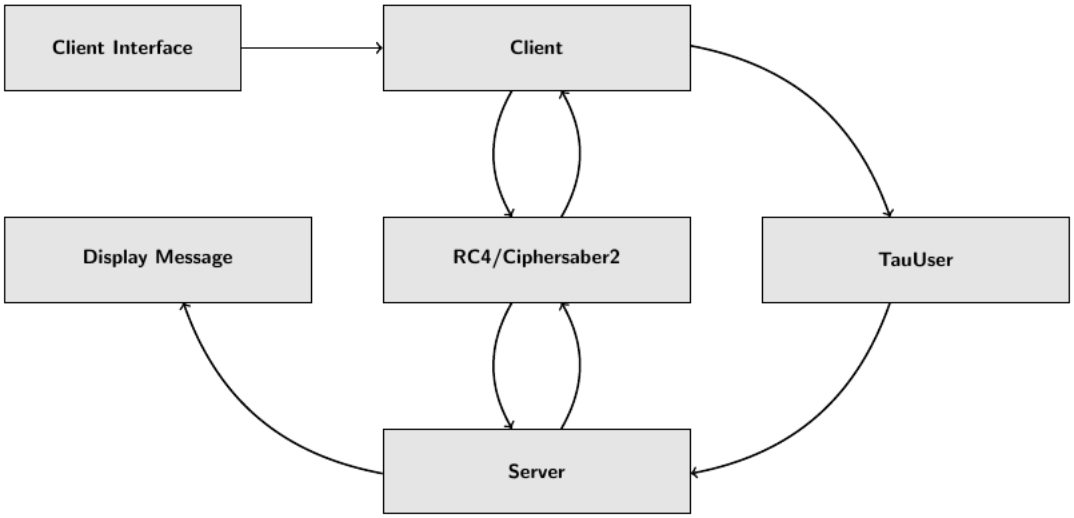

Figure 1: Simple TauNet implementaion design diagram

4 CONCLUSION

The TauNet design implemented is simple and functional and leaves plenty of room to be expanded on. This design should be considered as version one and will contain future updates. All the requirements have been met in accordance with the TauNet SRS and TauNet protocol.