Software Design Document

Version 1.0

December 08, 2015

TauNet

Jason Ritz

CS 300 Section 1 Fall 2015

Table of Contents

1. Introduction

1.1 Purpose

This software design document describes the system design of the TauNet system, including hardware and software requirements. It is intended to describe the implementation of the TauNet system.

2. Platform:

2.1 Hardware:

Raspberry Pi 2B

Network adaptor

8 Gb Micro SD

Keyboard

Mouse

Monitor

2.2 Operating System:

Raspbian Linux (a version of Debian Linux)

2.3 Software Required:

Python 2.7

3. TauNet Software Overview

The TauNet software consists of two python 2.7 files, the TauNet_Server.py, and the TauNet_Client.py. The TauNet_Server.py program is responsible for receiving incoming messages and displaying them on the screen. The TauNet_Client.py program is used to send messages to other TauNet users. Each of these programs are intended to be run in their own terminals.

3.1 TauNet_Server.py

When started, the TauNet server creates a TCP socket that binds to port 6283. It then listens for incoming connections. When an incoming connection is received it accepts the connection and saves the incoming string (up to 1024 bytes) and continues to listen for connections (up to 5 simultaneous connections). This string is then decrypted by removing the first 10 bytes of the string and using them as the IV for the decrypting process. Once the string is decrypted it displays the message on the screen and loops back to wait for a new incoming connection.

TauNet server start:

1. Create new socket
2. Bind socket to port 6283
3. Listen for incoming connection
4. Accept incoming connection
5. Receive incoming message up to 1024 bytes
6. Remove 10 byte IV from start of message
7. Send message and IV to decrypting function
8. Print message to screen
9. Go to step 3

3.2 TauNet _Client.py

When started, the TauNet client asks the user for their user name which is stored to be used for header information. Then the program asks the user for the user name of the person they would like to send a message to and waits for user input. Once the user enters the name of the TauNet user to send the message to it is stored for use in the header. Then the user is prompted to enter a message. Once entered the message is sent through the add header function where a header is added to the front of the message. The program then creates a random 10 byte IV and sends the message and the IV to the encrypt function. Once the message is encrypted the IV is attached to the front of the message. The program then looks up the IP address of the user to send to from the address book and creates a socket and connects to the user's IP on port 6283. The message is then sent and the socket is closed. The program then loops back to the start.

1. Get user name from user
2. Get user name to send to
3. Get message to send
4. Add header to message
5. Create random 10 byte IV
6. Encrypt message using key and IV
7. Add IV to front of message
8. Look up user IP from address book
9. Create TCP socket
10. Connect to other user
11. Send message
12. Close socket
13. Go to step 2

3.3 Message prep function process

The message prep function takes the message to be sent, and the to and from user names and prepares them to be sent to the encryption process by creating a single string of text that includes the header and message. It then creates a random 10 byte IV to be sent to the encryption process with the text message.

The header is in the form:

version: 0.2\r\n
to = to_user \r\n\r\n
sender = from_user \r\n

1. Populate the header with the to/from usernames
2. Attach message to be sent to the header creating a single plain text string
3. Generate 10 random bytes for the IV
4. Send plain text string and IV to the encryption function

3.4 Encrypt function process

Encryption and decryption are done using the RC4 cipher saber method as described in the ciphersaber document. TauNet uses a random IV of 10 bytes as a salt for the encryption process and a private predetermined key. This section covers how the cipher text is created from the plain text string created in the message prep function and the IV.

The encrypt function takes the message to be sent and the IV created during the message prep function as arguments. It then adds the IV to the end of the key and runs the cipher saber algorithm (described in the ciphersaber document). The IV is then attached to the front of the resulting cipher text, and the resulting string is then returned.

1. Add IV to end of the key string
2. Send message and key through the cipher saber algorithm
3. Attach IV to front of resulting cipher text
4. Return cipher text

3.5 Decrypt function process

Encryption and decryption are done using the RC4 cipher saber method as described in the ciphersaber document. TauNet uses a random IV of 10 bytes as a salt for the encryption process and a private predetermined key. This section covers how the received cipher text is converted into the message that will be displayed on screen.

The decrypt function takes the encrypted cipher text received from another TauNet and removes the first 10 bytes from the string to use as the IV. It then attaches the IV to the end of the key and uses the resulting key to decrypt the message by running the cipher saber algorithm (described in the ciphersaber document) on the cipher text. When the message is decrypted the function forgets the IV and returns the plain text string.

1. Remove the first 10 bytes of the string to be used as the IV
2. Attach the IV to the end of the Key
3. Send message and key through the cipher saber algorithm
4. Discard the IV
5. Return the plain text