Software Design Document

TauNet Version 1.0

December 08, 2015

CS 300 Elements of Software Engineering

Table of Contents

1.0. Introduction

1.1. Purpose

The purpose of this document is to present a detailed description of the implementation of the TauNet communication network.

1.2. Scope

This communications network was made according to Bart Massey's requirement specification. The software is based on the Python 3 programming language. The main libraries used for this software is the socket and thread Python libraries. This enables the ability to send bytes and strings through a TCP internet connection using multiple threads.

1.3. Glossary

Term	Definition
Developer	Person assisting with creation of the TauNet.
Encryption	A process of translating a message, called the Plaintext, into an encoded message, called the Ciphertext.
TauNet	Encrypted texting network between Raspberry Pi computers.
Raspberry Pi	A price efficient, credit-card sized computer.
RC4	A symmetric key cipher and bite- oriented algorithm that encrypts computer files and disks.
User	Person using a Raspberry Pi to communicate to other users through TauNet.

1.4. References

IEEE. *IEEE Std 29148-2011 Systems and software engineering – Life cycle processes – Requirements engineering*. IEEE Computer Society, 2011.

TauNet Protocol Version 0

1.5. Overview

The purpose of this document is to present a detailed description how the TauNet network was created using the Python programming language. The next section will give a brief overview on how the software operates. The following sections after will explain the architectural design of the software, the relationships between each class, and an in-depth description on how each class operates.

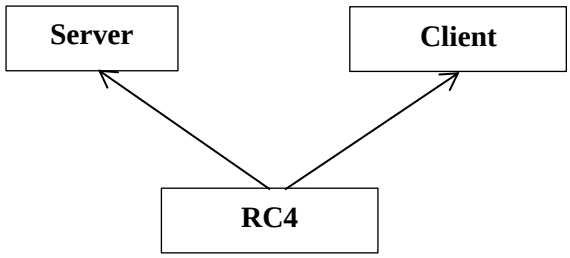
2.0. System Overview

Python was chosen as the primary programming language because it is cross-platform and is the built-in language for the Raspberry Pi 2. Three main files are used: a listening socket, a client socket, and an RC4 file. The listening socket creates multiple threads so it can connect and receive multiple clients. This allows the user to receive messages from more than one person at a time. The client socket allows to connect to other listening sockets and send messages to their respective servers. The RC4 file is implemented to handle the RC4 encryption algorithm and the CipherSaber2 implementation. The CipherSaber2 allows the decryption of incoming byte and string files while the encryption allows to send secret messages from a client to its server.

3.0. System Architecture

3.1. Architectural Design

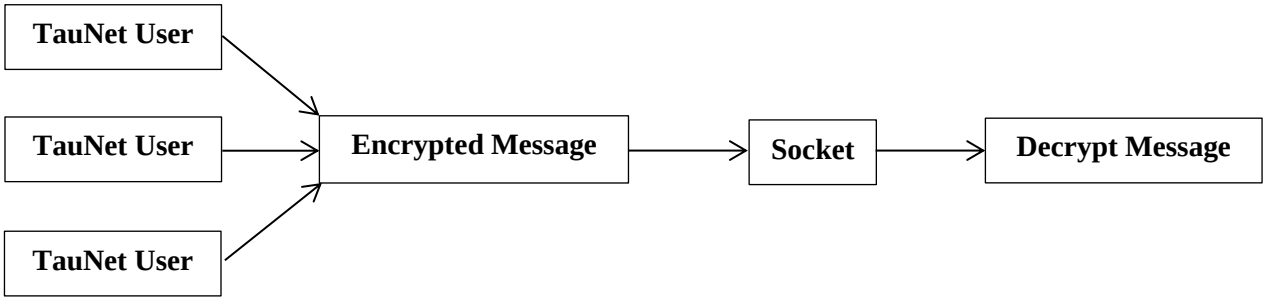
The main file in this software will be the Client and the main support will be the RC4. The user will mainly interact with the Client file that consists mostly of a socket that will connect to a listening server and send an encrypted message. A table of usernames with their respective IP addresses will be provided for ease of use in the Client file. The Server file will only be listening for other clients to connect and displaying a message. The Client and Server file will both use the RC4 file to encrypt and decrypt. The RC4 never outputs anything but does the main computation to create cipher texts for sending and receiving messages.



3.2. Decomposition Description

The first order for the Client process is to create a socket. A username is then chosen to return the IP address the socket will connect to. If connected the Client will proceed to send a message that includes the TauNet version number, username of the client, and username of the recipient. Before the message is sent, the message will be encrypted by RC4. This entails how the Client process works.

The first order for the Server process is to create a socket. The socket will then bind to the user's IP address. This socket will only listen and receive encrypted files. On each receive, all message will be decrypted by RC4. The server file mostly differs from the Client because it has only one function to listen and it will be threaded. This will allow multiple clients to message the user's server.rs to send and receive messages.



The RC4 file's main purpose is to create and read ciphertexts. Every time a message is sent or received, functions of the RC4 file will be called. Its other purposes are to convert bytes, bytearrays, and strings among each other to output and return the appropriate message.

3.3. Design Rationale

While building this project I had to choose to either use a multiple execution implementation or a single file execution. I chose the multiple file execution to make the program simpler and easier to read. One executable is the Server side that will just listen for all input, while the other (Client) will send output. The tradeoffs for the multiple file execution is that the user must keep the Server file and Client file open at the same time to communicate with someone. While the single file execution needs just one file, it needs to implement threading that may be difficult to use.

Another design was to create a way to implement group chats and save messages after each session. The tradeoff for this design would be increasing the potential of an information leak due to saved messages. Thus, all messages are deleted after each session.

4.0. Data Description

The data structures in this program is very simplistic. All input and output are strings in the Client and Server process. The RC4 file converts these strings into bytes, byte arrays, integer arrays, and strings appropriately. A table of users is read from a csv file.

5.0. Human Interface Design

5.1. Overview of User Interface

The user will have to execute two processes: Server and Client. Once Server processes, a notification will pop up for every connection made to the user and messages will automatically be displayed. For the Client process, the user is prompted to choose a username to send a message to. Once chosen, a connection may be made if the chosen user is online. Afterwards, the user is prompted again to create a message to send until the connection is closed with the message “exit”.

5.2 Screen Images

```
***SERVER***
[AWAITING CONNECTION]
```

```
[USERNAME]        [HOST ADDRESS]        [PORT]
['tdulcet', 'tealdulcet.ddns.net', '6283']
['patter5', 'megmurry.ddns.net', '6283']
['relsqui', 'cupcake.chiliahedron.com', '6283']
['pyrrh', 'souffle.chiliahedron.com', '6283']
['aarong', 'inspignia.ddns.net', '6283']
['mancat', 'inchworm.mindtax.net', '6283']
['dom', 'mahan.mindtax.net', '6283']
['brodie', 'solar.noip.me', '6283']
['paolo2', 'pirr.ddns.net', '6283']
['mrme', 'mrme.ddns.net', '6283']
['manpreet20', 'manpreet.ddns.net', '6283']
['huyng90', 'huyng90.ddns.net', '6283']
['rhatchet', 'pi.arenjae.com', '6283']
['natreed', 'natreed.ddns.net', '6283']
['leng', 'junkgrave.com', '6283']
['aldridge85', 'csjosh.ddns.net', '6283']
['etsai', 'etsai.ddns.net', '6283']
['jbucklin', 'jbucklin.ddns.net', '6283']
['echo', 'barton.cs.pdx.edu', '6283']
['echo1', 'po8.org', '6283']
['echo2', 'drupal.svcs.cs.pdx.edu', '6283']
['mantron', 'mantron.ddns.net', '6283']
['jojen2', 'itsjonnyjyo.ddns.net', '6283']
['cort', 'wyeast.ddns.net', '6283']
['rubear', '131.252.211.242', '6283']
['castlez', 'castlez.ddns.net', '6283']
['quanah', 'quanah.duckdns.org', '6283']
['mattTighe', 'matthewtighe.ddns.net', '6283']
['jbrophy', 'jbrophy.ddns.net', '6283']
['ph6', 'paulhuynh94.ddns.net', '6283']
['zixuchen', 'zixuchenlorch.ddns.net', '6283']
['mouacher', 'mouacher.ddns.net', '6283']
['stripes416', 'mattst88.com', '6283']
['mohl', 'mohl.ddns.net', '6283']
['pbn', 'pbn.ddns.net', '6283']
['hyena', 'hyena.ddns.net', '6283']
['Minion', 'minion.mindtax.net', '6283']
['gtdrakeley', '71.59.140.217', '6283']
['po8', 'bartfan.po8.org', '6283']
['waspxor', 'TBD', '6283']

INPUT BY HOST ADDRESS(1) OR IP(2)?

INPUT:
```

```
***SERVER***
[AWAITING CONNECTION]
CONNECTION FROM:  ('20.10.10.110', 55361)

RECEIVED:
version: 0.2
from: Jason
to: Friend

I am good how about you?

RECEIVED:
version: 0.2
from: Jason
to: Friend

Yea! Totally ready! How about you?

RECEIVED:
version: 0.2
from: Jason
to: Friend

Bye!
```

```
['dom', 'mahan.mindtax.net', '6283']
['brodie', 'solar.noip.me', '6283']
['paolo2', 'pirr.ddns.net', '6283']
['mrme', 'mrme.ddns.net', '6283']
['manpreet20', 'manpreet.ddns.net', '6283']
['huyng90', 'huyng90.ddns.net', '6283']
['rhatchet', 'pi.arenjae.com', '6283']
['natreed', 'natreed.ddns.net', '6283']
['leng', 'junkgrave.com', '6283']
['aldridge85', 'csjosh.ddns.net', '6283']
['etsai', 'etsai.ddns.net', '6283']
['jbucklin', 'jbucklin.ddns.net', '6283']
['echo', 'barton.cs.pdx.edu', '6283']
['echo1', 'po8.org', '6283']
['echo2', 'drupal.svcs.cs.pdx.edu', '6283']
['mantron', 'mantron.ddns.net', '6283']
['jojen2', 'itsjonnyjyo.ddns.net', '6283']
['cort', 'wyeast.ddns.net', '6283']
['rubear', '131.252.211.242', '6283']
['castlez', 'castlez.ddns.net', '6283']
['quanah', 'quanah.duckdns.org', '6283']
['mattTighe', 'matthewtighe.ddns.net', '6283']
['jbrophy', 'jbrophy.ddns.net', '6283']
['ph6', 'paulhuynh94.ddns.net', '6283']
['zixuchen', 'zixuchenlorch.ddns.net', '6283']
['mouacher', 'mouacher.ddns.net', '6283']
['stripes416', 'mattst88.com', '6283']
['mohl', 'mohl.ddns.net', '6283']
['pbn', 'pbn.ddns.net', '6283']
['hyena', 'hyena.ddns.net', '6283']
['Minion', 'minion.mindtax.net', '6283']
['gtdrakeley', '71.59.140.217', '6283']
['po8', 'bartfan.po8.org', '6283']
['waspxor', 'TBD', '6283']

INPUT BY HOST ADDRESS(1) OR IP(2)?

INPUT: 2

INPUT IP: 20.10.10.110
CONNECTED TO: 20.10.10.110 6283

SEND: Hello how are you?

SEND: Are you ready for the winter break?

SEND: I am going to eat!

SEND: Bye!

SEND: exit
[DISCONNECTED]
```

```
***SERVER***
[AWAITING CONNECTION]
CONNECTION FROM:  ('20.10.10.108', 53039)

RECEIVED:
version: 0.2
from: Jason
to: John

Hello how are you?

RECEIVED:
version: 0.2
from: Jason
to: John

Are you ready for the winter break?

RECEIVED:
version: 0.2
from: Jason
to: John

I am going to eat!

RECEIVED:
version: 0.2
from: Jason
to: John

Bye!
```

```
['jojen2', 'itsjonnyjyo.ddns.net', '6283']
['cort', 'wyeast.ddns.net', '6283']
['rubear', '131.252.211.242', '6283']
['castlez', 'castlez.ddns.net', '6283']
['quanah', 'quanah.duckdns.org', '6283']
['mattTighe', 'matthewtighe.ddns.net', '6283']
['jbrophy', 'jbrophy.ddns.net', '6283']
['ph6', 'paulhuynh94.ddns.net', '6283']
['zixuchen', 'zixuchenlorch.ddns.net', '6283']
['mouacher', 'mouacher.ddns.net', '6283']
['stripes416', 'mattst88.com', '6283']
['mohl', 'mohl.ddns.net', '6283']
['pbn', 'pbn.ddns.net', '6283']
['hyena', 'hyena.ddns.net', '6283']
['Minion', 'minion.mindtax.net', '6283']
['gtdrakeley', '71.59.140.217', '6283']
['po8', 'bartfan.po8.org', '6283']
['waspxor', 'TBD', '6283']

INPUT BY HOST ADDRESS(1) OR IP(2)?

INPUT: 2

INPUT IP: 20.10.10.108
CONNECTED TO: 20.10.10.108 6283

SEND: I am good how about you?

SEND: Yea! Totally ready! How about you?

SEND: Bye!

SEND: exit
[DISCONNECTED]
```